UNIVERSIDAD CARLOS III DE MADRID

TRABAJO FIN DE GRADO

CONECTIVIDAD BLUETOOTH CON SENSORES BIOMÉTRICOS EN DISPOSITIVOS MÓVILES

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Carlos Sánchez Redondo

Tutor: Raúl Sánchez Reíllo

Leganés, 15 de septiembre de 2014

Resumen

En la actualidad existe una tendencia generalizada hacia la tecnología móvil. Se intenta reducir el tamaño y la dependencia externa de nuestros dispositivos. Debido a esto, en los últimos años se ha visto un gran auge de nuevos sistemas operativos móviles como Android o iOS, y la creación y el desarrollo de dispositivos móviles como móviles inteligente o tabletas. Hoy en día un elevado porcentaje de la población dispone de acceso a un dispositivo móvil, siendo para muchos un dispositivo indispensable para su vida diaria. Del mismo modo, ciertas tecnologías inalámbricas como WiFi o Bluetooth han tomado mucha más importancia, al tratar de minimizar el uso de cables y medios físicos para la transmisión de datos.

Por otro lado, la seguridad personal y la identificación están tomando mayor importancia. Sistemas como la identificación por contraseña o número pin comienzan a decaer en favor de tecnologías biométricas tales como la identificación por huella dactilar o identificación facial. La identificación por huella dactilar lleva décadas siendo usada para confirmar la identidad de una persona, gracias a ventajas como su universalidad, unicidad y facilidad de captura, pero es en la actualidad cuando se está haciendo accesible a la mayoría de la población con la evolución de las tecnologías móviles. El hecho de simplemente poner un dedo o hacer una foto para desbloquear un teléfono, realizar un pago o identificarse, supone una gran comodidad para las personas, siendo un sistema normalmente más seguro que otros métodos de identificación.

Este trabajo de fin de grado nace con el objetivo de desarrollar un sistema con esa tecnología. Un sistema de identificación biométrica utilizando un sensor de huella dactilar que se capaz de comunicarse de forma inalámbrica, por medio de Bluetooth, con un dispositivo Android. Se ha creado una aplicación Android basada en el estándar BioAPI para orientación de objetos (ISO/IEC 30106), un estándar cuya función es permitir la comunicación de aplicaciones software con un amplio número de tecnologías biométricas de una manera común. La aplicación puede realizar las operaciones básicas de un sistema biométrico: captura, almacenamiento y verificación de información biométrica.

Este documento introduce la tecnología utilizada y muestra el proceso de desarrollo del sistema, la implementación en OO BioAPI y la creación de la aplicación final.

Abstract

Nowadays there is a general trend towards mobile technology, attempting to reduce the size of the devices and its external dependence. Because of this, in the last years there has been a big upswing of new operating systems such as Android or iOS and the creation and development of mobile devices such as smartphones or tablets. Today, a high percentage of the population has access to a mobile device, in some cases being essential for daily life. Similarly, certain wireless technologies such as Wi-Fi or Bluetooth have reached higher significance, trying to minimize the use of wires and physical media for data transmission.

On the other hand, personal security and identification are becoming more important. Systems such as password or pin code identification begin to wane in favor of biometric technologies like fingerprint or facial identification. Fingerprint identification has been used to confirm the identity of an individual for years due to its advantages such as universality, unicity and ease of capture, but it is now when it is becoming accessible for the majority of the population thanks to the evolution of mobile technologies. The fact of just putting a finger or taking a picture to unlock a phone, make a payment or identify oneself is a big convenience for people, being this typically safer than other identification methods.

This Bachelor Thesis is born with the objective of developing a system with this technology. A biometric identification system using a fingerprint sensor capable of wireless communication via Bluetooth with an Android device. An Android application has been created based on the BioAPI object-oriented standard (ISO/IEC 30106), a standard which function is to allow the communication of software applications with a broad number of biometric technologies in a common way. The application can perform the basic operations of a biometric system: capture, archive and verification of biometric information.

This paper introduces the technology used and shows the development process of the system, the OO BioAPI implementation and the creation of the final application.

Índice

Índice de Figuras

Índice de Tablas

Listado de Acrónimos

ADT	*Android Development Tool* (Herramienta de desarrollo Android)
API	*Application Programming Interface* (Interfaz de programación de aplicaciones)
BDB	*Biometric Data Block* (Bloque de datos biométricos)
BioAPI	*Biometric Application Programming Interface* (Interfaz de programación de aplicación biométrica)
BIR	*Biometric Identification Record* (Registro de identificación biométrica)
BSP	*Biometric Service Provider* (Proveedor de servicios biométricos)
IDE	*Integrated Development Enviroment* (Entorno de desarrollo integrado)
JVM	*Java Virtual Machine* (Máquina virtual java)
MAC	*Media Access Control* (Control de acceso al medio)
SDK	*Software Development Kit* (Kit de desarrollo de software)
SPI	*Service Provider Interface* (Interfaz de prooverdores de servicios)
TFG	Trabajo de Fin de Grado
UUID	*Universally Unique Identifier* (Identificador universalmente único)

1 Introducción

En este documento se presentará un TFG cuyo objetivo es el desarrollo de un sistema de identificación biométrica basado en BioAPI utilizando un sensor de huella dactilar que sea capaz de comunicarse de forma inalámbrica, por medio de Bluetooth, con un dispositivo Android.

A lo largo del trabajo se introducirán las tecnologías empleadas, la elección de dispositivos y el desarrollo del sistema.

1.1 Motivación y objetivos

Una de las razones por las que se decidió realizar este TFG fue el interés por el aprendizaje de programación en el sistema operativo Android. Android ha crecido notablemente en importancia en los últimos años, convirtiéndose en uno de los sistemas más destacados en la actualidad. El poder aprender cómo funciona el sistema y cómo es el desarrollo de una aplicación de principio a fin era muy atractivo. Otra razón es la curiosidad por los sistemas de identificación. Tema muy interesante no tratado durante la titulación cursada. El trabajo une ambas tecnologías, permitiendo crear un sistema completo de principio a fin. De este modo, se pretende crear un sistema biométrico funcional de principio a fin, tomando todas las decisiones de diseño e implementación.

El hecho de no tener experiencia con estas tecnologías era una motivación, ya que permitía al alumno adquirir nuevos conocimientos en tecnologías actuales, que podría ser útil en su futuro profesional.

El objetivo del trabajo es el desarrollo de un sistema de identificación biométrica basado en BioAPI utilizando un sensor de huella dactilar que sea capaz de comunicarse de forma inalámbrica, por medio de Bluetooth, con un dispositivo Android.

Este objetivo general se puede desglosar en varios objetivos específicos:

- Aprendizaje de programación en Android y desarrollo de aplicación de ejemplo: la plataforma elegida para la conectividad con sensores biométricos es Android. Una plataforma muy extendida actualmente, libre y con mucha documentación, que la hace idónea para la tarea.
- Búsqueda de un sensor de huella dactilar Bluetooth compatible: encontrar un sensor de huella dactilar compatible con la plataforma Android y con un SDK (*Software Development Kit*) en JAVA capaz de comunicarse mediante Bluetooth.
- Aprendizaje sobre reconocimiento biométrico y BioAPI: obtener los conocimientos básicos de biometría, en concreto huella dactilar, y aprender la estructura y funcionamiento de BioAPI, un estándar utilizado para el software de captura y comparación de información biométrica.
- Creación de un sistema biométrico capaz de capturar, almacenar y verificar huellas dactilares. Para ello se desarrollará una aplicación Android que se comunique con el sensor de huella Bluetooth.

1.2 Entornos socio-económico y marco regulador

Los sistemas de identificación biométrica, en particular los basados en huella dactilar, llevan décadas siendo indispensables en diversos ámbitos como el control de acceso o la identificación personal. Durante este tiempo han estado en constante evolución, pero es en la actualidad cuando, gracias a nuevos dispositivos como los móviles inteligentes, esta tecnología se está convirtiendo en atractiva y accesible para el uso personal además de aumentar su uso en ámbitos profesionales.

Este crecimiento se debe a que la identificación por huella dactilar es una manera cómoda y segura de confirmar la identidad de una persona, y los nuevos dispositivos móviles son capaces de aprovecharla. De esta manera, cada vez más compañías incluyen sensores de huella dactilar en sus dispositivos, sistema que comienza a reemplazar métodos como claves de seguridad, siendo la identificación biométrica más segura. El crecimiento de las tecnologías móviles va ligado al desarrollo de la comunicación inalámbrica, que también ha ganado mucha importancia en los últimos años, y gracias a la que los dispositivos no necesitan conexión directa con sensores, añadiendo comodidad.

Gracias a la capacidad de los nuevos dispositivos, no solo se incluyen sensores de huella dactilar en ellos, sino que pueden hacer uso de otras tecnologías biométricas tales como identificación facial o por iris. Estos usos de la identificación biométrica tienen el objetivo de mejorar el día a día de los usuarios, mejorando su seguridad al mismo tiempo.

En cuanto al marco regulador, es importante destacar que la información biométrica es considerada información personal. Todo proyecto y producto que utilice autenticación biométrica debe seguir los principios de la directiva 2009/136/CE del parlamento europeo y del consejo sobre protección de datos personales [1]. En el caso de España, la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal (BOE de 14-12-1999) [2].

1.3 Estructura del documento

En este documento en primer lugar se realizará una introducción a las tecnologías utilizadas. En el primer capítulo se hablará de la biometría, particularizando en la huella dactilar (apartado 2, Biometría). A continuación se introducirá el estándar BioAPI (apartado 3, BioAPI). Y se explicarán las herramientas de desarrollo, siendo éstas el lenguaje de programación Java y el sistema operativo Android, así como el entorno de desarrollo que será utilizado (apartado 4, Herramientas de desarrollo).

Una vez explicadas estas tecnologías, en los apartados 5 y 6 se expondrá el problema y su solución, explicando en detalle las decisiones tomadas y el proceso de desarrollo. También se mostrará el funcionamiento de la aplicación final y su comportamiento.

Por último, se expondrán las conclusiones obtenidas y se expondrán posibles líneas futuras de desarrollo en el apartado 7.

2 Biometría

En este apartado se introducirán brevemente las nociones básicas de la biometría. Se expondrá su historia y las características de los sistemas biométricos. A continuación se explicarán las particularidades de la modalidad utilizada, la huella dactilar.

2.1 Historia de la biometría

La biometría es el estudio de métodos para el reconocimiento de forma única y automática de seres humanos mediante sus características físicas, biológicas o de comportamiento.

El origen de la biometría se remonta hasta el antiguo Egipto, donde se usaba la huella dactilar para confirmar la identidad de las personas. Hay evidencias del uso de identificación por huella dactilar en China en el siglo XIV, en las que se dejaba la huella dactilar marcada sobre arcilla.

En Europa se comenzó a usar en el siglo XIX con fines policiales el sistema antropométrico [3] que funcionaba midiendo de forma precisa diversos parámetros de cada sujeto y registrando características particulares. Éste fue el primer sistema preciso, ampliamente utilizado de forma científica, convirtiendo a la biometría en un ámbito de estudio. Para combatir los fallos de este sistema, más adelante se comenzó a usar la huella dactilar como principal método de identificación, de una manera muy similar al sistema utilizado en China en el siglo XIV.

La biometría ha evolucionado desde entonces, mejorando los sistemas existentes y desarrollando nuevas tecnologías. Avances como sistemas de detección de “dedo vivo” y nuevas tecnologías basadas en identificación facial o por iris han sido desarrollados para solucionar problemas tales como la dificultad de captar la huella de una persona o la posibilidad de engañar al sistema. [4]

2.2 Sistema biométrico

Para realizar la identificación de una persona se puede utilizar prácticamente cualquier característica biológica o de comportamiento. La huella dactilar, los rasgos faciales, el ADN, así como la manera de andar o de realizar una firma son métodos utilizados para la identificación.

Normalmente las características biológicas son consideradas más exactas, ya que el comportamiento al realizar una acción se puede replicar, siendo menos fiable.

A la hora de juzgar una técnica biométrica, son muchos los parámetros que hay que considerar, de los que se pueden destacar los siguientes [5]:

- **Universalidad**: cualquier sujeto puede pertenecer al sistema.
- **Unicidad**: dos muestras de sujetos diferentes no pueden tener las mismas características.
- **Estabilidad**: las características del usuario tienen que permanecer inalteradas.
- **Facilidad de captura**: mecanismo de toma de muestras sencillo.
- **Robustez frente al fraude**: necesidad de detección de usuario vivo o amenazado.
- **Aceptación por parte de los usuarios**: que el uso del sistema no provoque rechazo a los usuarios.
- **Coste**: bajo coste de los equipos e implantación
- **Grado de servicio**: llegar a un compromiso entre la seguridad, el coste y el servicio.

Atendiendo a esas características, no existe una técnica de identificación única. Cada técnica tiene sus ventajas e inconvenientes. Técnicas que ofrecen unos resultados excelentes, no pueden ser usadas en muchos entornos debido a su dificultad al obtener las muestras o su alto coste. Por otro lado, técnicas que ofrecen un nivel de seguridad inferior, por otras razones pueden ser más fácilmente empleadas en determinados entornos, al ser más importantes las ventajas que proporcionan. Por ejemplo, la identificación por ADN garantiza un 100% de éxito en la identificación, pero es una técnica muy costosa y necesita mucho tiempo para obtener resultados. Sin embargo otras técnicas como la identificación por huella dactilar, que es menos exacta, son mucho más fáciles de realizar y menos costosas.

2.3 Etapas de un sistema biométrico

Para desarrollar un sistema de identificación biométrica se utiliza un esquema independiente de la modalidad de reconocimiento biométrico empleado. (Figura 1)

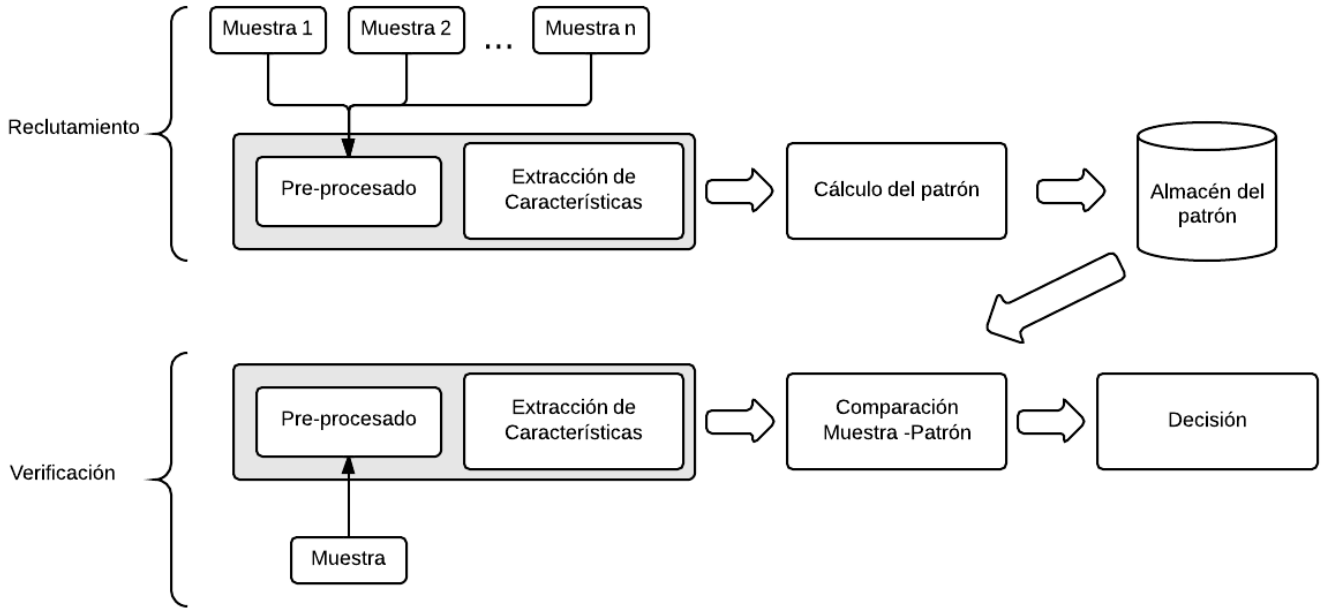

Figura 1: Etapas de un sistema biométrico

Las dos fases principales de cualquier sistema biométrico son [5]:

- **Reclutamiento:** en esta primera fase, la información biométrica del usuario es obtenida y procesada. Una vez se dispone de la información, se obtiene un patrón del usuario, esto es el conjunto de datos que caracterizan a ese usuario en concreto. En el caso de que se capturen más de una muestra del mismo usuario, el patrón suele ser una media de las características obtenidas. El proceso se hace de forma supervisada para minimizar los errores y enseñar al usuario el funcionamiento del sistema.
- **Verificación:** al igual que en el reclutamiento, se obtiene y procesa la información biométrica obtenida, pero en este caso la información biométrica se compara con el patrón almacenado, determinado el éxito o el fracaso de la operación.

La verificación de la información biométrica no es definitiva, ya que aunque las muestras sean del mismo individuo, siempre existen pequeñas diferencias entre ellas debido al error de los instrumentos de captura o variaciones de la muestra. Por ello, no se puede obtener un resultado positivo o negativo, sino que hay que trabajar con porcentajes de semejanza. De este modo, la verificación de dos muestras devolverá

un valor de comparación, que hay que evaluar para determinar la posibilidad de que las muestras sean del mismo individuo.

2.3.1 Métricas de evaluación

Los sistemas biométricos determinan el resultado de una comparación basándose en un umbral de decisión. Este umbral determina con qué nivel de igualdad dos muestras son consideradas del mismo individuo. Variarlo puede dar lugar a errores, generando falsos positivos (las muestras no son del mismo usuario pero las considera así) y falsos negativos (no reconoce que las muestras sean del mismo usuario aunque lo sean).

Para determinar la precisión de un sistema biométrico se suelen usar dos tasas de error [6]:

- **FAR**, Tasa de falsa aceptación (*False Acceptance Rate*): mide la cantidad de errores que comete el sistema cuando reconoce muestras de distintos usuarios como pertenecientes al mismo, generando falsos positivos. También se conoce como FMR (*False Match Rate*).
- **FRR**, Tasa de falso rechazo (*False Rejection Rate*): mide la cantidad de errores que comete el sistema cuando no reconoce huellas pertenecientes al mismo usuario, considerándolas pertenecientes a distintos usuarios. Falsos negativos. También se conoce como FNMR (*False Non Match Rate*).

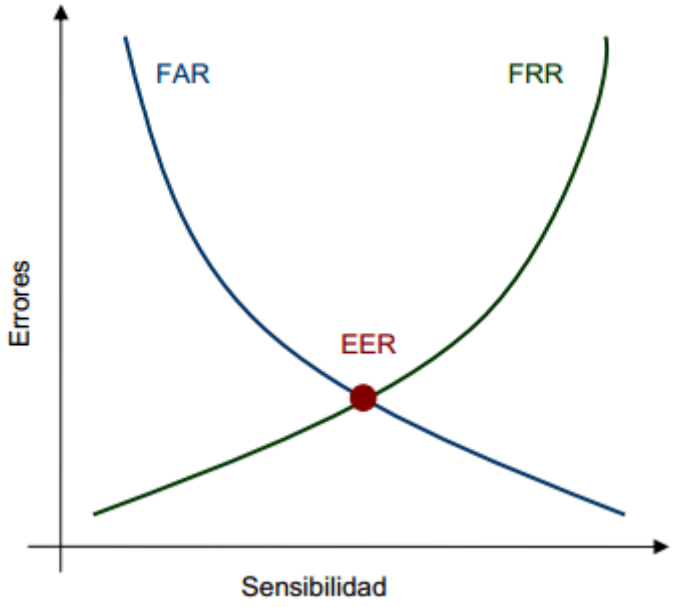

Figura 2: Tasas de error de un sistema biométrico [6]

Como se aprecia en la figura 2, el FAR y FRR varían de forma inversa, cuanto más aumenta uno más disminuye el otro. El EER (*Equal Error Rate*) determina el punto en el que ambas tasas de error tienen el mismo valor, pero en un sistema real no suele utilizarse, ya que, dependiendo del sistema, se suele tender a minimizar una de las tasas.

Estas tasas de error determinan la calidad del sistema biométrico, pero hay que tener en cuenta que evalúan las capacidades de los algoritmos de identificación del sistema, sin tener en cuenta otras características del sistema como la calidad de los dispositivos de captura.

2.4 Huella dactilar

Como se ha establecido en los objetivos de este trabajo, se trabajará con la modalidad de identificación biométrica de huella dactilar. En este apartado se describirán las características de la huella dactilar y se introducirán los conceptos básicos del funcionamiento de su procesado e identificación.

2.4.1 Características de la huella dactilar

Una huella dactilar es la impresión o marca que se produce al hacer contacto con las crestas papilares de un dedo sobre una superficie. Las huellas dactilares son detalladas, únicas, difíciles de alterar y duran toda la vida del dueño, por lo que son una característica ideal para la identificación biométrica de las personas.

Como se ha comentado, las huellas dactilares fueron el primer sistema de identificación biométrica utilizado.

El análisis de una huella dactilar con el propósito de su comparación atiende a diversas características de cada huella [7]:

- **Patrón**: es la manera en la que están dispuestas las crestas de la huella. Existen tres formas básicas, arcos, presillas o lazos y verticilos o espirales, tal como se ve en la figura 3.

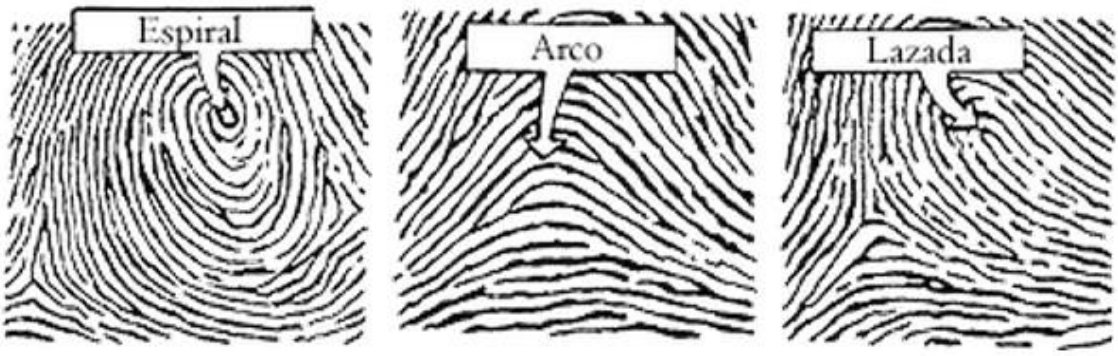

Figura 3: Patrones de huella dactilar [7]

- **Minucias**: son características únicas que se encuentra en las crestas y en las formas. Entre ellas se pueden destacar algunas como los finales de cresta, las bifurcaciones y las crestas pequeñas o puntos. En la figura 4 se ve un ejemplo de bifurcación y final de cresta.

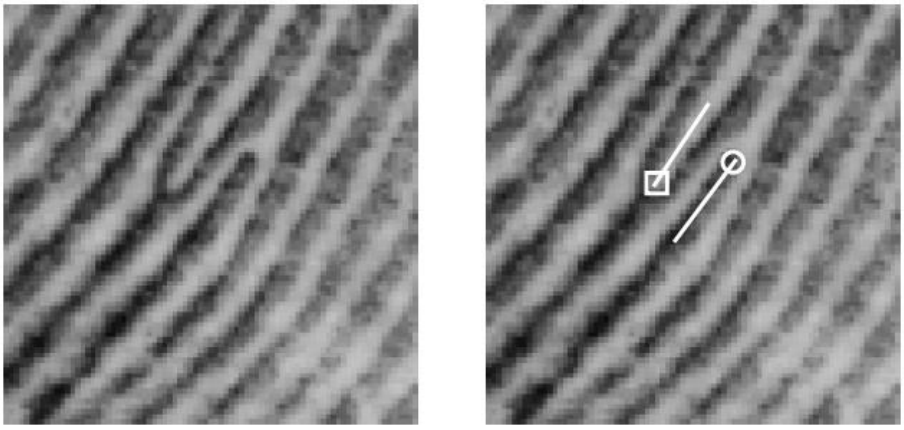

Figura 4: Minucias [18]

Gracias a estas características, la huella dactilar cumple varios parámetros necesarios para ser un buen sistema de identificación biométrico. Es **universal**, ya que, exceptuando casos inusuales, todas las personas poseen una huella dactilar. Es **única,** cada individuo posee una huella dactilar distinta a la de los demás. Es **estable,** a lo largo de la vida de una persona la huella no varía. Es **fácil de capturar,** el método de captura es muy simple y no es invasivo.

2.4.2 Captura de huella

La captura de una huella dactilar se puede realizar sobre un formato físico como cartulina o papel, o de forma digital mediante sensores específicos.

Para realizar la captura sobre un formato físico se realiza una impresión dactilar, ésta consiste en posar la huella impregnada en tinta sobre un papel o cartulina generalmente.

Para realizar la captura de forma digital, se utilizan sensores de huella. Estos sensores son dispositivos electrónicos diseñados para realizar la captura. Este tipo de captura es llamada captura en vivo. Como se ha explicado en el apartado 2.3, una vez el sensor toma una imagen de la huella, generalmente la huella es procesada para crear un patrón (reclutamiento), que será utilizado para realizar el proceso de verificación.

Los tipos de sensores más utilizados son [7]:

- **Ópticos**: capturan la huella dactilar por medio de luz visible. En esencia es una cámara digital que captura una foto de la huella directamente. Los inconvenientes de este tipo de sensores son el que pueden ser engañados por una imagen de una huella si no tienen sistemas de detección de “dedo vivo” y que la captura puede ser mala si hay suciedad o arañazos en el sensor y si el dedo está sucio. Estos sensores pueden capturar imágenes sin tener contacto con el dedo.
- **Ultrasónicos**: usan ondas de sonido de alta frecuencia para penetrar en la epidermis. Estas ondas son generadas por transductores piezoeléctricos. Una vez los ultrasonidos llegan a la dermis del dedo (la dermis y la epidermis tienen el mismo patrón de huella), son reflejados y recibidos por el sensor, formando la imagen de la huella. Estos sensores no necesitan que el dedo y el sensor estén limpios, y no pueden ser engañados por una imagen.
- **Capacitivos**: usan principios asociados con la capacidad eléctrica para capturar la imagen. La superficie del sensor está cubierta con un *array* de placas, que funcionan como una de las placas paralelas de un condensador, la epidermis es utilizada como dieléctrico y la dermis, que es conductora, funciona como la otra placa del condensador. La capacidad entre el sensor y el dedo es menor cuando detecta una cresta y mayor cuanto más espacio hay entre el sensor y el dedo. De este modo se puede obtener una imagen clara de la huella dactilar. Este tipo de sensores son susceptibles de sufrir descargas electrostáticas. Si el dedo tiene callos o cicatrices no capturan correctamente la imagen. La humedad, la grasa y el polvo pueden afectar a su funcionamiento.

2.4.3 Algoritmos de comparación

Debido a las numerosas aplicaciones posibles para el reconocimiento de huella dactilar y a su implantación en ámbitos policiales, así como de control de acceso desde hace décadas, se han desarrollado numerosos algoritmos de comparación.

Las técnicas de comparación se pueden dividir en estos subgrupos [6]:

- **Comparación de minucias**: Las huellas son inicialmente procesadas para obtener la posición de sus minucias. Una vez procesadas, se compara posición de las minucias, de esta manera si un gran porcentaje de minucias coinciden, la huella tiene mayor probabilidad de ser la misma. Es el método más utilizado.
- **Comparación de crestas**: Nacido como alternativa a la comparación de minucias, comparan ciertas características de las crestas, como por ejemplo su grosor o sus poros.
- **Correlación**: Realizan la correlación de los pixeles de las imágenes para comprobar el grado de similitud entre ellas.

En el apartado 6 se explicarán los algoritmos utilizados para este trabajo.

3 *Biometric Aplication Programming Interface* (BioAPI)

El estándar BioAPI es una interfaz de programación creada para permitir la comunicación de aplicaciones software con un amplio número de tecnologías biométricas de una manera común.

En BioAPI se definen interfaces entre módulos que permiten la integración de software de distintos fabricantes. De este modo, su principal objetivo es proporcionar una manera estandarizada de crear sistemas compatibles con diferentes desarrolladores y fabricantes, permitiendo su intercambio.

3.1 Historia

En el campo de la biometría, históricamente no han existido estándares. Cada fabricante desarrollaba su sistema adaptado para sus dispositivos, de forma incompatible con sistemas de otros fabricantes. Si se deseaba cambiar o ampliar un sistema biométrico existente con dispositivos de otro proveedor, era necesario rediseñar el sistema por completo. Esto dificultaba el crecimiento del sector biométrico, ya que invertir en él era arriesgado.

Para solucionar este problema en 1998 se formó el consorcio BioAPI, formado por varias empresas del ámbito tecnológico, con el objetivo de desarrollar un estándar biométrico [8].

Se quería crear una interfaz de programación de aplicaciones (API) y una interfaz de proveedores de servicios (SPI) para definir una base común a partir de la que los fabricantes y desarrolladores pudiesen trabajar. De este modo, los sistemas desarrollados a partir de BioAPI podrían operar entre sí y ser intercambiables.

La primera especificación de BioAPI fue lanzada en el año 2000. Era compatible con otros estándares que estaban saliendo en la misma época, para aumentar la compatibilidad y facilidad de creación de sistemas biométricos. En 2003 dicha especificación fue ofrecida al recientemente creado comité internacional de estandarización ISO/IEC JTC1/SC37, el cual terminó publicando una nueva versión, más avanzada, en 2005, bajo el número de estándar ISO/IEC 19784.

La especificación realizada, tanto previamente, como bajo la norma ISO/IEC 19784, estaba realizada en lenguaje ANSI C, por lo que ha tenido muchos problemas de adaptación a la industria, ya que en la época en la que fue publicada, casi todas las aplicaciones se desarrollaban en lenguajes con orientación a objetos. Esto llevó a que se iniciasen en 2009 los trabajos parar crear una nueva versión BioAPI ajustada a este tipo de lenguajes, es decir, lo que se conoce como *Object Oriented* BioAPI, y que

será publicado en 2015 con el código ISO/IEC 30106. En los desarrollos de dicha especificación trabaja muy intensamente el Grupo Universitario de Tecnologías de Identificación (GUTI) de la Universidad Carlos III de Madrid, grupo en el que se ha realizado el presente TFG. Este trabajo es un paso más en la depuración de dicho estándar, así como en la depuración de la documentación, permitiendo que desarrolladores sin conocimientos previos del estándar, puedan desarrollar nuevas aplicaciones basadas en ISO/IEC 30106 en un breve espacio de tiempo.

3.2 Arquitectura

BioAPI está estructurado por niveles de una manera jerárquica, como se observa en la figura 5.

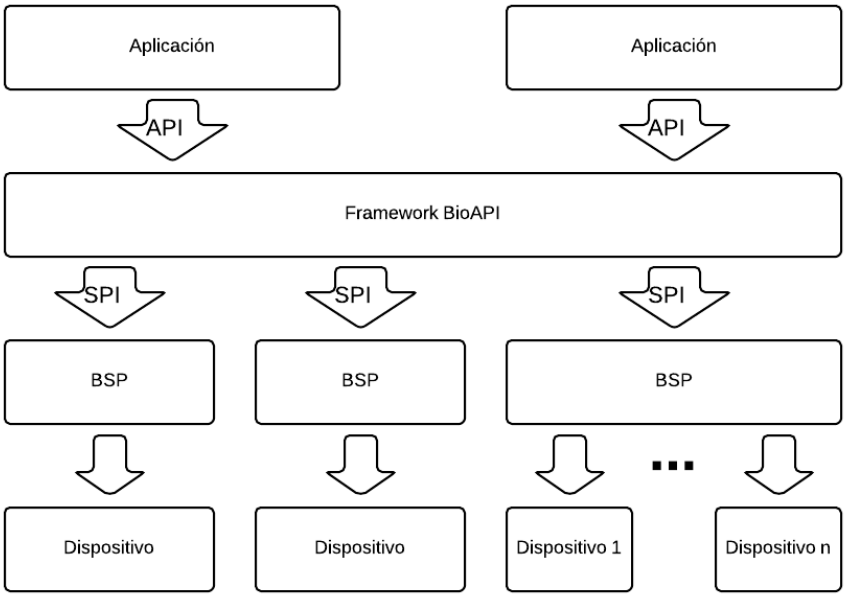

Figura 5: Arquitectura de BioAPI

En el nivel más bajo se sitúan los dispositivos biométricos, esto son los sensores, que recibirán la información biométrica para introducirla en el sistema.

Por encima de los dispositivos se encuentran los BSP (*Biometric Service Providers*). En ellos se engloban los dispositivos, así como diversos módulos utilizados para realizar funciones tales como la captura, el procesado, el almacenamiento y la comparación de la información biométrica provista por los dispositivos. Esos servicios podrán ser provistos a la aplicación utilizando además un BFP (*Biometric Function Provider*), que puede gestionar las funciones de los diversos BSP.

En el siguiente nivel se encuentra el *framework*. El *framework* comunica las aplicaciones con los BSP, permitiendo el acceso a todas las funciones provistas por ellos. Para la comunicación con los BSP, situados un nivel inferior, el f*ramework* utiliza la interfaz SPI.

En el nivel superior, se encentran las aplicaciones. Las aplicaciones funcionan como interfaz gráfica para que los usuarios puedan comunicarse con el sistema. Para invocar las funciones que se encuentran en los BSP, las aplicaciones utilizan el API para comunicarse con el f*ramework*, el que a su vez obtendrá las funciones del BSP deseado. [9]

3.3 *Biometric Service Provider* (BSP)

El BSP, Proveedor de Servicio Biométrico, da acceso a las operaciones biométricas. Es un elemento de mucha importancia, ya que proporciona todas las operaciones que se pueden realizar en el sistema biométrico.

Los BSP son los elementos que permiten que una aplicación o un sistema desarrollado con BioAPI sean estándar, ya que disponen de la misma interfaz independientemente del sistema biométrico utilizado. De esta manera, el objetivo de un BSP es permitir utilizar una única aplicación para distintos tipos de sistemas biométricos, diferentes sensores y diferentes algoritmos.

Por ejemplo, desde la aplicación se podría hacer una llamada al método “capture”, cuya función es obtener información biométrica de un sensor, y gracias al BSP este método es indiferente del tipo de sensor, tanto si en un sensor de huella dactilar, como una cámara para identificación facial o una captura de firma, el método será el mismo. El BSP se encargará de identificar el tipo de sensor y las funciones necesarias para realizar correctamente la captura.

Cabe destacar dos métodos esenciales del BSP:

- ***Enrol***: se encarga de gestionar completamente el registro de un nuevo usuario en el sistema. Engloba todas las llamadas a funciones necesarias para completar el proceso.
- ***Verify***: se encarga de realizar las llamadas necesarias cuando se quiere realizar una verificación. Al llamarlo, realizará una captura, la procesará y la comparará con un (o varios) BIR almacenados en la base de datos. Como resultado devolverá el FMR de las comparaciones.

En el apartado 6.3.5 se profundizará en el funcionamiento de estos métodos.

Los métodos que se encuentran englobados en un BSP se agrupan en diferentes subgrupos dependiendo de su función y de su procedencia. Estos grupos son las unidades.

3.3.1 Unidades

Existen cuatro tipos de unidades que se encuentran dentro de un BSP: archivo, procesado, comparación y sensor. Cada una de ellas tiene su función específica, y se pueden incluir dentro de un BSP tantas como sea necesario. El disponer de unidades añade modularidad al sistema, permitiendo no incluir unidades cuando no sean necesarias y la adición de éstas sin alterar el funcionamiento del resto ni de BSP.

En el apartado 6.3 se explicarán en detalle las distintas unidades y métodos utilizados en el trabajo. En este apartado se introducirán las unidades para la comprensión de su funcionamiento.

- **Unidad de Sensor:**

La unidad de sensor gestionará las funciones necesarias para el correcto funcionamiento de un sensor biométrico. En principio cada sensor utilizado por el sistema dispondrá de su propia unidad, la que contendrá las funciones necesarias para poder comunicarse con el sensor.

El método más relevante de la unidad de sensor es *capture*. Este método permitirá establecer comunicación con el sensor para la obtención de la información biométrica, devolviendo un BIR (registro de información biométrica, explicado en el apartado 3.4). Por ello, dentro de la unidad se encontrarán las funciones utilizadas para la inicialización del sensor, el establecimiento de comunicación con él y las funciones necesarias para capturar la información biométrica.

De esta forma, cuando el BSP llama al método *capture* de una unidad de sensor, independientemente de su tipo o sus características, siempre recibirá un BIR con la información biométrica capturada del sensor.

- **Unidad de Procesado:**

En esta unidad se encuentran las funciones utilizadas para el procesamiento de la información biométrica. Estas funciones utilizarán los algoritmos de procesado necesarios para la correcta comparación de la información biométrica. Puede existir más de una unidad de procesado, dependiendo del tipo de sistema biométrico o incluso distintos algoritmos para un mismo tipo de sistema.

El método más relevante de la unidad será el método *process*. Este método recibe la información biométrica previamente capturada en forma de BIR, la procesa y devuelve otro BIR con la información procesada. En el caso de una huella dactilar, como ejemplo podría recibir una imagen bruta de huella y devolvería un vector con la posición de las minucias.

Al igual que con la unidad de sensor, cuando la función *process* sea invocada por el BSP, dependiendo del BIR provisto se utilizará la unidad necesaria, sin tener que especificar el tipo de sistema biométrico.

- **Unidad de Comparación:**

En ella se encuentran los métodos para realizar la comparación de la información biométrica. Al igual que la unidad de procesado, las funciones de esta unidad utilizarán distintos algoritmos para comparar la información biométrica almacenada en varios BIR. La información biométrica recibida por esta unidad ha sido procesada antes (en la mayor parte de los casos).

Su método más relevante el método *verify*. Su función es aplicar los algoritmos necesarios para la comparación de dos BIR distintos, devolviendo un valor de correlación entre las dos muestras. Normalmente la comparación se realiza entre un BIR recién capturado y un BIR previamente almacenado, siendo el almacenado un patrón.

Como con las demás unidades, pueden existir distintas unidades de comparación para distintas modalidades de identificación y distintos algoritmos, y el método puede ser invocado indistintamente sin tener en cuenta la modalidad.

- **Unidad de Archivo:**

Almacena los BIR capturados y gestiona las bases de datos utilizadas. Principalmente se utilizará para complementar a la unidad de comparación, que necesita los BIR almacenados para compararlos con las nuevas capturas.

En esta unidad se encuentran los métodos utilizados para crear bases de datos, introducir nuevos miembros y gestionarlos.

3.4 *Biometric Identification Record* (BIR)

Un BIR funciona como una unidad de información biométrica. Como se ha mencionado en el apartado 3.3, puede contener información procesada y sin

procesar, así como funcionar como un método de comunicación entre distintos niveles de BioAPI.

Un BIR dispone de una serie de campos, unos obligatorios y otros opcionales, que se rellenarán dependiendo de la utilización que se le dará.

El BIR está definido por la norma ISO/IEC 19785, en donde se define su estructura.

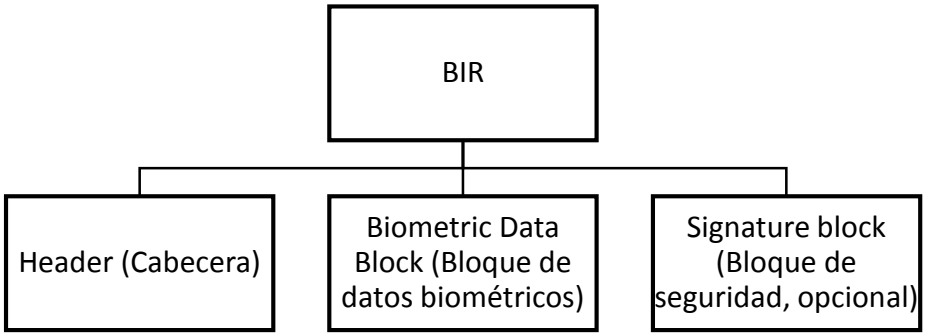

Figura 6: BIR

Como se muestra en la figura 6, el BIR dispone de una cabecera, un bloque de datos biométricos (BDB) y un bloque de seguridad.

En la cabecera se incluyen todas las características del BIR. Entre ellas el proveedor del BIR, el tipo, si se encuentra encriptado o la versión de la norma utilizada.

En el BDB se almacena la información biométrica, ya haya sido procesada o no. De este modo, cuando se captura información biométrica, ésta se almacena en el apartado BDB de un BIR. Este BIR se podría procesar por medio del método *process* en la unidad de procesado, generando un nuevo BIR con la imagen procesada.

Para poder almacenar correctamente un BIR en una base de datos, es necesario convertirlo a un formato que el dispositivo sea capaz de almacenar, por ello el BIR dispone de funciones, utilizadas por el método *storeBIR* de la unidad de archivo que permiten convertir el BIR a un *array* de bytes, siendo mucho más fácil de tratar y almacenar.

4 Herramientas de desarrollo

Como se ha establecido en los objetivos del trabajo, éste se desarrollará sobre la plataforma Android.

En este apartado se introducirán los conceptos básicos de Android, así como del lenguaje de programación Java, que es utilizado para el desarrollo de aplicaciones Android. Adicionalmente se hablará del entorno de desarrollo utilizado para el trabajo.

4.1 Java

El lenguaje de programación Java fue desarrollado en 1995 por Sun Microsistems. Su sintaxis es similar a la de C y C++, pero carece de tantas herramientas a bajo nivel como ellos. Java se puede ejecutar en cualquier JVM (*Java Virtual Machine*) independientemente del sistema operativo o la arquitectura del sistema donde se ejecuta.

Java es un lenguaje de programación de propósito general, orientado a objetos y basado en clases, que fue diseñado para depender lo mínimo posible de elementos externos. De esta manera, su objetivo es permitir que las aplicaciones desarrolladas en Java se puedan ejecutar en cualquier dispositivo sin necesidad de recompilarlo, siendo interpretado en tiempo de ejecución por la máquina virtual.

Estas características hacen de Java un lenguaje simple y sencillo de usar a la expensa de perder utilidades, y el hecho de que una aplicación sea interpretada la convierte en más lenta que otras aplicaciones desarrolladas en lenguajes como C o C++.

En la actualidad, Java es uno de los lenguajes de programación más utilizados, como se muestra en la figura 7 según el índice TIOBE de popularidad de lenguajes de programación. [10]

Aug 2014	Aug 2013	Change	Programming Language	Ratings	Change
1	2	^	C	16.401%	+0.43%
2	1	v	Java	14.984%	-0.99%
3	4	^	Objective-C	9.552%	+1.47%

Figura 7: Índice de popularidad de lenguajes de programación [10]

En el desarrollo del proyecto se explicarán las principales clases y funciones utilizadas.

4.2 Android

Android es un sistema operativo basado en el *kernel* de Linux que fue diseñado principalmente para su uso en móviles inteligentes, tabletas y dispositivos con pantalla táctil. [11]

Fue diseñado inicialmente por Android Inc., empresa que compró Google en 2005. Android se presentó en 2007 junto a la fundación *Open Handset Alliance*, un consorcio de compañías del ámbito tecnológico formado para mejorar los sistemas abiertos para dispositivos móviles.

El primer móvil que portaba el sistema operativo Android fue el *HTC Dream*, saliendo a la venta en octubre del año 2008.

Actualmente, el sistema operativo se encuentra en un gran rango de aparatos electrónicos, entre ellos teléfonos inteligentes, tabletas, ordenadores portátiles, televisores o relojes de pulsera.

4.2.1 Arquitectura

El sistema operativo Android dispone de una serie de componentes principales, tal como se ve en la figura 8:

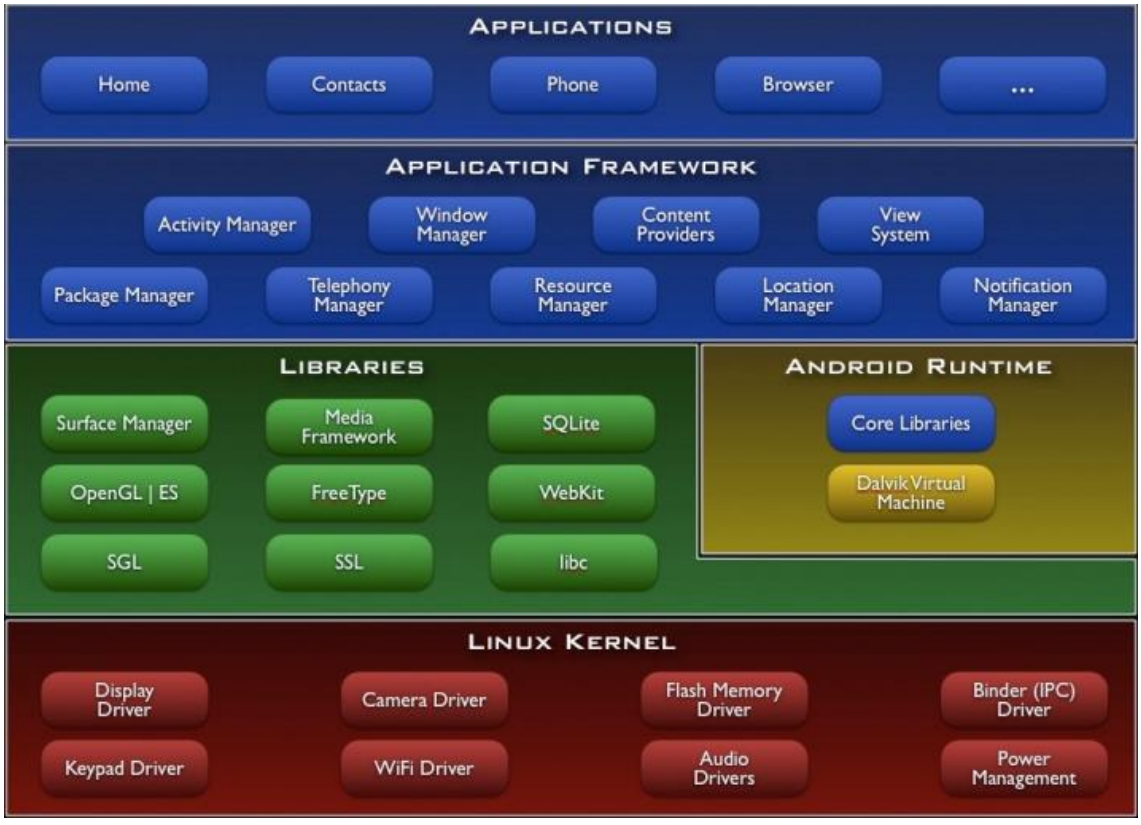

Figura 8: Arquitectura de Android [11]

- **Aplicaciones**: al igual que en sistemas operativos como Microsoft Windows o Ubuntu, en el sistema Android se pueden instalar aplicaciones con distintos propósitos, como por ejemplo un navegador web, un procesador de textos o un videojuego. Las aplicaciones le dan una increíble versatilidad a los dispositivos, permitiendo aprovechar al máximo su potencial. Todas las aplicaciones Android están desarrolladas en el lenguaje de programación Java.
- ***Framework* de las Aplicaciones**: gracias a este marco de trabajo, los desarrolladores tienen acceso a diversos API, con lo que pueden utilizar los recursos provistos por el sistema operativo y el hardware disponible en el dispositivo.
- **Bibliotecas**: Android dispone de diversas bibliotecas escritas en C para expandir y mejorar la funcionalidad del *kernel* Linux. Entre ellas se encuentran bibliotecas de gráficos, *SQLite* para bases de datos y WebKit para ciertos navegadores web.
- ***Runtime* Android**: un set de bibliotecas base, las que proporcionan la mayoría de las funciones del lenguaje de programación Java. Las aplicaciones Java necesitan una máquina virtual para ser ejecutadas, en el *Runtime* Android se encuentra la máquina virtual *Dalvik*, que ejecuta las aplicaciones.
- **Kernel Linux**: para los servicios básicos del sistema, Android depende de Linux. Estos servicios son la seguridad, la gestión de la memoria y los procesos, los drivers de hardware, etc. Además, el *kernel* funciona como una capa de abstracción entre el software y el hardware, estableciendo la comunicación entre ellos.

4.2.2 Componentes de una aplicación

Existen una serie de elementos clave que son imprescindibles para el desarrollo de cualquier aplicación Android. A continuación se introducirán algunos de los más importantes.

- ***Activity***: son el componente visual de la aplicación, formando su interfaz. Son las “pantallas” que ve el usuario y con las cual interactúa.
- ***View***: son todos los elementos de la interfaz de usuario de la aplicación. Por ejemplo, los botones, los campos de texto o las imágenes.
- ***Service***: son componentes sin interfaz gráfica, procesos que se ejecutan en segundo plano sin necesidad de interacción con el usuario.
- ***Intent***: la forma de comunicación entre distintos elementos de una aplicación. Representa la intención de realizar alguna acción, y se utiliza cada vez que se necesite lanzar una actividad o servicio, comunicarse con un servicio o enviar un anuncio tipo *broadcast*.

- ***Broadcast Receiver***: recibe los anuncios tipo *broadcas*" lanzados. Éstos pueden ser originados por el sistema (batería baja, llamada entrante…) o por la aplicación.
- ***Content Provider***: permite a las aplicaciones comunicarse entre ellas, sin necesidad de acceder al sistema de ficheros comprometiendo su seguridad. Con este sistema se puede acceder a los datos de otras aplicaciones, como la lista de contactos o la galería, o proporcionar datos a otras aplicaciones.

4.2.3 Entorno de desarrollo

Los IDE son programas informáticos compuestos por un conjunto de herramientas de programación que permiten a los desarrolladores escribir, compilar, probar mediante una interfaz gráfica y depurar sus programas de una forma rápida y sencilla.

Para el desarrollo de aplicaciones Android existen diversos entornos de desarrollo, siendo los más utilizados:

- **Eclipse**:

Eclipse es un entorno de desarrollo integrado (IDE) multiplataforma y de código abierto. Eclipse soporta programación en diversos lenguajes como C, C++, Fortran o Phyton entre otros, pero su uso más extendido es con Java.

Eclipse dispone de un entorno de trabajo base, que se puede extender mediante diversos *plug-ins*, que permiten aumentar su funcionalidad y personalizar el entorno.

Para el desarrollo de este trabajo se ha utilizado Eclipse junto a su *plug-in* ADT (*Android Development Tools*), que adapta el entorno de trabajo para permitir desarrollar aplicaciones Android.

- **Android Studio:**

Otro entorno de desarrollo integrado capaz de desarrollar aplicaciones Android es Android Studio. Basado en el entorno IntelliJ IDEA, ha sido diseñado específicamente para desarrollar aplicaciones Android. El programa está actualmente en fase beta desde el año 2013, por lo que no dispone de todas las características y puede dar problemas.

El hecho de haber sido diseñado específicamente para Android es un beneficio, ya que elimina muchas características no utilizadas de otros entornos como Eclipse.

Durante el desarrollo de este trabajo se han utilizado ambos entornos, pero, debido al estado inestable y el poco tiempo que lleva disponible Android Studio (lo que influye en la cantidad de información y solución a problemas disponible), el trabajo final se ha desarrollado en Eclipse.

5 Diseño de la solución

En este apartado se planteará el problema propuesto y el objetivo del trabajo. Se mostrarán las diversas opciones de hardware y software investigadas y se comentarán las decisiones tomadas al respecto, así como las distintas decisiones con respecto a la estructura y el diseño de la solución.

5.1 Planteamiento del problema

Como se ha mencionado, el objetivo del trabajo es la conectividad Bluetooth con sensores biométricos en dispositivos móviles a través de estándar BioAPI. En particular, la realización de una interfaz de comunicación entre un sensor de huella dactilar Bluetooth y un dispositivo móvil Android, utilizando BioAPI Java.

La interfaz deberá ser capaz de comunicarse de forma inalámbrica con el sensor Bluetooth, recibir datos de él y procesarlos. Todo ello deberá hacerlo de una manera cómoda y accesible, con facilidad para el usuario final.

Desde la interfaz se podrá capturar una imagen con el sensor, procesarla y almacenarla siguiendo las etapas de un sistema biométrico explicadas en el apartado 2.3. También se podrán comparar nuevas capturas con las huellas almacenadas para realizar identificaciones.

Esta interfaz se implementará en un dispositivo Android, utilizando el estándar BioAPI explicado en el apartado 3.

5.2 Planteamiento de la solución

Una vez están claros los objetivos, la solución se desarrollará en los siguientes pasos:

- Búsqueda de un sensor de huella dactilar Bluetooth compatible: encontrar un sensor de huella dactilar compatible con la plataforma Android y con un SDK (*Software Development Kit*) en JAVA capaz de comunicarse mediante Bluetooth.
- Desarrollo de una aplicación capaz de comunicarse con el sensor vía Bluetooth y de recibir datos de él utilizando el SDK provisto por el fabricante.
- Implementación del SDK del sensor en BioAPI.

- Creación de una aplicación de ejemplo que muestre el funcionamiento completo del sistema biométrico utilizando BioAPI.

5.2.1 Búsqueda y elección del sensor

La mayoría de los móviles inteligentes no disponen de sensor de huella dactilar, por lo que se necesita disponer de un dispositivo externo. Este sensor deberá cumplir las siguientes características:

- **Conectividad Bluetooth**: deberá poder comunicar con el sensor de manera inalámbrica.
- **SDK compatible con Android**: para poder comunicarse con el sensor de forma apropiada, éste debe poseer un SDK escrito en Java que permita las operaciones necesarias para recibir la información que envía.

No existe necesidad de que el sensor sea compatible con otros dispositivos o sistemas operativos, así como otro tipo de conexiones. Tampoco hay restricciones entre los distintos tipos de sensor de huella dactilar.
El hecho de que el sensor sea inalámbrico añade comodidad para el usuario final y versatilidad a los posibles usos del sistema.
La función del sensor será capturar la información biométrica del usuario, para seguidamente transmitírsela al dispositivo Android, que realizará todo el procesado. El sensor no alterará la imagen ni la procesará de ningún modo.
Al ser unas características poco comunes, existen pocos dispositivos que se adapten a ellas en el mercado.

5.2.2 Desarrollo de aplicación con el SDK del sensor

Una vez en posesión del sensor, se necesitará comprobar que se puede realizar una comunicación fiable con él y se pueden recibir los datos deseados.

Se realizará una aplicación básica que se comunique con el sensor, utilizando para ello el SDK de éste.

Para el desarrollo de esa aplicación se necesitarán conocimientos de programación en la plataforma Android y en el lenguaje de programación Java explicados en el apartado 4. A su vez será necesario conocer el funcionamiento de las conexiones Bluetooth y su implementación en una aplicación Android.

Con esta primera aplicación se comprobará el funcionamiento del sensor, capturando las imágenes de huella dactilar captadas y enviándolas al dispositivo Android.

Esa aplicación será la base con la que se desarrollará la aplicación final, ya que una vez terminada, las funciones necesarias serán incluidas en un BSP de BioAPI (ver apartado 3.3), lo que permitirá su uso sin necesidad de utilizar el SDK directamente.

5.2.3 Implementación del sensor en BioAPI

Una vez se compruebe que el sensor funciona de manera deseada y se puedan recibir datos de él, se implementará en BioAPI.

Como se ha explicado en el apartado 3, el sistema se puede dividir en varios niveles como se observa en la figura 9:

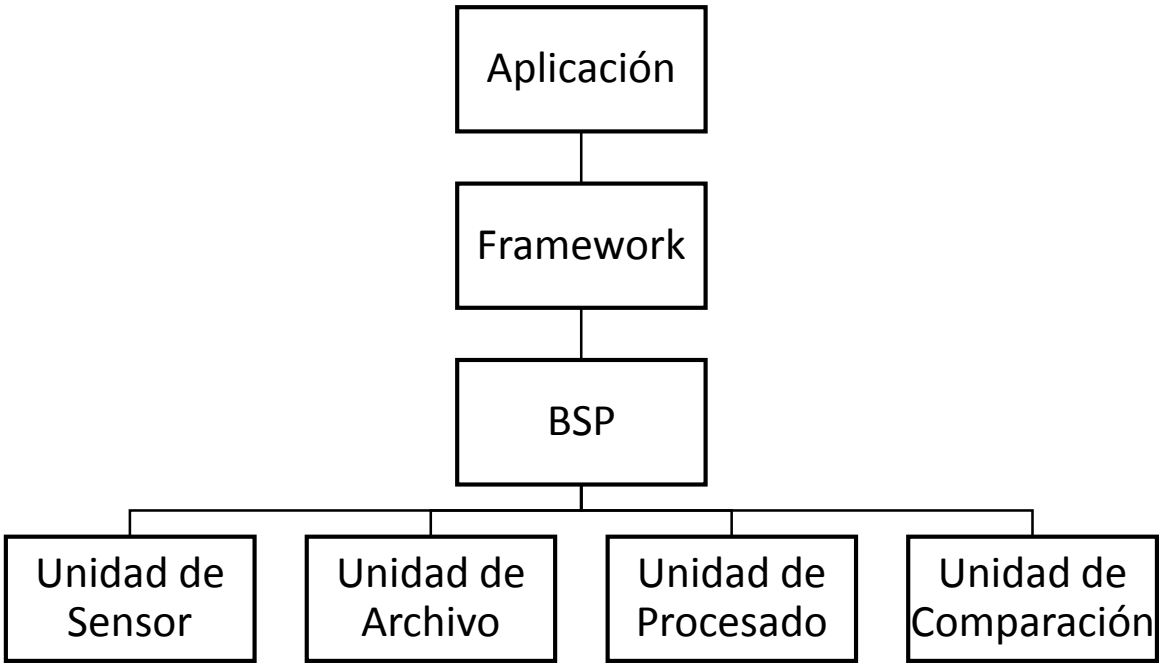

Figura 9: Estructura de BioAPI

- **Aplicación**: será la interfaz gráfica del sistema y su función consistirá en transmitir las acciones del usuario a los siguientes niveles, así como mostrarle la información deseada al usuario. También podrá realizar operaciones que no estén relacionadas con BioAPI.

- ***Framework***: es el elemento encargado de gestionar la comunicación entre la aplicación y los BSP, así como de controlar todos los elementos conectados a él, de esta manera se puede comprobar la disponibilidad de los diversos BSP por la aplicación, ya que ésta no tiene acceso directo a ellos.

- **BSP**: agrupa todas las unidades y gestiona la comunicación entre el *framework* y las unidades.

- **Unidades**: realizarán las operaciones necesarias para almacenar, procesar y comparar la información biométrica.

En este caso, no es necesario la utilización de un Framework, ya que sólo se trabajará con un BSP, siendo el *framework* una manera de simplificar la utilización de múltiples BSP. Sin embargo se ha decidido utilizarlo para mostrar un entorno BioAPI completo y permitir posibles ampliaciones futuras.

Se creará un BSP para englobar las unidades necesarias. Desde el BSP se podrán realizar operaciones tales como *enrol* (permite capturar y almacenar una huella en un BIR, así como información extra y posibilidad de procesarla) o *verify* (comparación de una huella dactilar capturado con una o varias almacenadas) sin tener que acceder directamente a las funciones de la unidades.

Para el objetivo propuesto, serán necesarios diversas unidades dentro del estándar BioAPI:

- **Sensor**: engloba todas las funciones necesarias para poder conectarse al sensor y recibir datos de él. Estos datos serán almacenados en un BIR.

- **Procesado**: recibe un BIR capturado y obtiene la información biométrica necesaria para poder comparar la huella dactilar con otras.

- **Archivo**: almacenamiento de los BIRs capturados. Este almacenamiento se realiza por medio de una base de datos SQL.

- **Comparación**: recibe la información procesada y la compara con los BIRs almacenados.

En concreto, será necesario crear una unidad de sensor específica para el sensor elegido, donde se encuentren todas las funciones usadas del SDK, y dispuesto de tal manera que no sea necesario utilizar el SDK del sensor para poder recibir datos de él, sino que se puedan realizar todas las operaciones pertinentes desde la unidad de sensor.

5.2.4 Desarrollo de la aplicación final

Una vez el sensor esté implementado en BioAPI junto al resto del BSP y el framework, se creará una aplicación que incluya las funciones principales.

Como se ha mencionado previamente, la aplicación permitirá realizar las etapas de un sistema biométrico explicadas en el apartado 2.3, la aplicación permitirá realizar el reclutamiento y la verificación de muestras biométricas. Para ello, la aplicación tendrá una serie de características:

- Conectarse con el sensor de forma inalámbrica, utilizando la tecnología Bluetooth.
- Realizar una captura de huella para procesarla y almacenarla (reclutamiento), recibiendo la información del sensor.
- Realizar una verificación de huella, esto es, comparar una captura con la base de datos almacenada y devolver una lista de posibles coincidencias (verificación).
- Control de las entradas de la base de datos, permitiendo ver información sobre éstas y borrarlas si es necesario.

En la figura 10 se muestra el diagrama general de la aplicación.

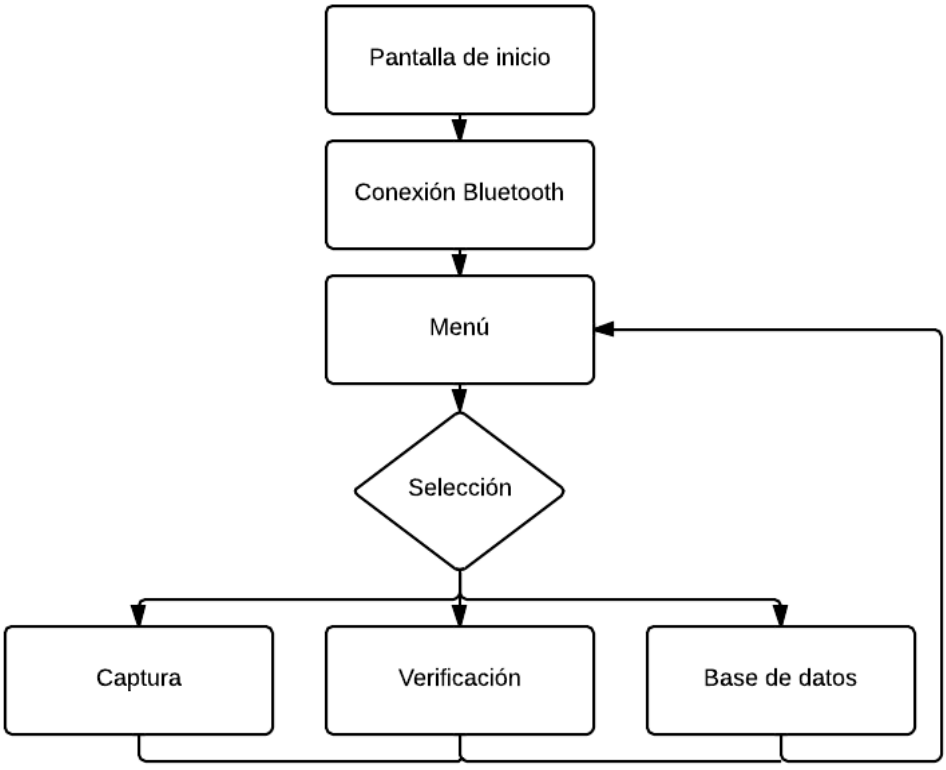

Figura 10: Diagrama general de la aplicación

6 Desarrollo

Una vez se ha planteado la solución, se entrará en detalle en el proceso de desarrollo del trabajo.

6.1 Búsqueda y elección del sensor

En primer lugar se necesitaba encontrar un sensor de huella dactilar adecuado para el trabajo.

Como se ha apuntado anteriormente, el sensor debía tener conectividad Bluetooth y debía disponer de un SDK compatible con Android.
No existía necesidad de que el sensor fuese compatible con otros dispositivos o sistemas operativos, así como otro tipo de conexiones. Tampoco había restricciones entre los distintos tipos de sensor de huella dactilar.

Eran unas características poco comunes en este tipo de dispositivos, ya que normalmente disponen únicamente de conexión vía USB. Por esta razón no existen muchos modelos en el mercado.

Se realizó una búsqueda preliminar para encontrar un número de dispositivos con esas características. Se buscó información referente al precio del dispositivo, las características del envío, la disponibilidad del SDK y la existencia de ejemplos de uso. Las opciones más destacadas fueron:

- **Pingan** (Lector de tarjetas RFID y sensor de huella dactilar):

Figura 11: Sensor Pingan [12]

-Precio: 125,96 dólares.

-Envío gratuito de 1 a 10 días.

-SDK: gratuito y código abierto. Soporta windows, windows mobile y Android.

-Sin ejemplos de uso.

Sensor barato y aparentemente adecuado para el trabajo, pero con muy poca información al respecto y sin ejemplos de funcionamiento.

- **Funtronic:**

Figura 12: Sensor Funtronic [13]

-Precio: 549 dólares Envío de 1 a 3 días.

-SDK: 950 dólares, se compran por separado los de mac, windows y linux/android.

-Existen varios ejemplos descargables en la página web.

Un dispositivo con mucha información sobre él y con características idóneas para el trabajo. Además dispone de más características como ser capaz de funcionar independientemente gracias a su propia interfaz, pero esto no es necesario para el trabajo. El mayor inconveniente de este dispositivo es su elevado precio comparado con los demás.

- **SMUFS-BT RAW:**

Figura 13: Sensor SMUFS-BT RAW [14]

-Precio: no disponible.

-SDK: multiplataforma con ejemplos en Java, C#, C y C++.

-Ejemplos en todos los lenguajes.

El dispositivo es aparentemente adecuado para el trabajo, pero la falta de información respecto al precio y el SDK es un inconveniente.

- **Bluefin:**

Figura 14 – Sensor BlueFin [15]

-Precio: 720 dólares, con el SDK incluido y la documentación, incluyendo los gastos de envío. 300 dólares al ser estudiante.

-SDK: incluido en el precio del sensor, escrito en Java y compatible con Android.

-Varios ejemplos y demostraciones.

Con varios ejemplos y con las características idóneas para el trabajo. El precio no es muy elevado y el sensor aparenta haber sido probado y utilizado para diversos proyectos.

El sensor elegido fue el **sensor BlueFin**, debido a sus características y las demostraciones encontradas de su funcionamiento.

Este dispositivo dispone del sensor "*Upek TCS1 TouchChip Silicon Fingerprint Sensor*", un sensor de silicio que usa la tecnología de capacitancia activa (apartado 2.4.2) [16]. Dispone de una superficie de 18.0 mm por 12.8 mm y genera imágenes de 256x360 pixeles.

El sensor BlueFin dispone de cifrado AES (*Advanced Encryption Standard*) de 256 bits, lo que aumenta la seguridad del dispositivo. El sensor es inalámbrico, por lo que dispone de una batería de 3,7 voltios y 1500 miliamperios recargable. [17]

6.2 Desarrollo de la aplicación con el SDK del sensor

Una vez se disponía del sensor sobre el que se iba a trabajar, se necesitaba comprobar su funcionamiento y familiarizarse con el SDK. Para ello, se decidió realizar una pequeña aplicación Android capaz de comunicarse con el sensor y de recibir imágenes suyas.

Antes de poder realizar la aplicación, se necesitaban conocimientos de programación en la plataforma Android y del funcionamiento de la conectividad Bluetooth.

Para poder realizar el trabajo se necesitaban conocimientos básicos de programación en Java y Android.

Se invirtió tiempo en familiarizarse con los entornos de desarrollo Eclipse y Android Studio. Como se ha mencionado en el apartado 4.2.3, se han utilizado ambos entornos, pero para el desarrollo de la aplicación final se ha utilizado Eclipse debido a su mayor estabilidad y documentación.

También era necesario familiarizarse con la API de desarrollo Android, para ello se crearon varias aplicaciones para conocer el funcionamiento básico de Android y las aplicaciones.

Esto permitió aprender el método de desarrollo y ganar confianza en el uso de las herramientas.

6.2.1 Estudio y aprendizaje de conectividad Bluetooth en dispositivos Android

Para la conectividad del sensor con el dispositivo Android es necesaria una conexión vía Bluetooth. Para que la conexión fuese posible es necesario desarrollar una serie de funciones para poder tener acceso al sistema Bluetooth integrado en el dispositivo Android y para establecer una comunicación con el sensor.

La plataforma Android provee de soporte para la pila de comunicación Bluetooth (*Bluetooth Network Stack*), que permite a un dispositivo a compartir de forma inalámbrica datos con otro dispositivo Bluetooth. El *framework* de la aplicación provee de acceso a las funcionalidades Bluetooth a través de las API de Bluetooth Android. Estas API permiten realizar conexiones punto a punto con otros dispositivos de manera inalámbrica.

Para poder realizar una conexión Bluetooth, existen cuatro pasos que hay que realizar. En primer lugar preparar el Bluetooth del dispositivo, esto es activarlo y comprobar que funciona correctamente. A continuación hay que encontrar un dispositivo al que conectarse. Después se realiza la conexión y para finalizar se gestiona esa conexión.

Para poder conectarse con cualquier dispositivo Bluetooth, únicamente se necesita la dirección MAC de ese dispositivo. Con esa dirección MAC se puede crear un socket y establecer un canal de comunicación entre los dispositivos. Esta dirección MAC se puede conocer de antemano, almacenándola en la aplicación o la memoria del teléfono, o se puede adquirir de los dispositivos visibles. En este trabajo se escaneará en busca de dispositivos y se le permitirá al usuario elegir a qué dispositivo desea conectarse.

Hay una serie de clases importantes que se utilizan para realizar la conexión:

- ***BluetoothAdapter***: Es el “adaptador Bluetooth” del dispositivo Android. Básicamente es la radio que emite y recibe los datos vía Bluetooth. Esta clase es necesaria para poder utilizar el Bluetooth del dispositivo, representa la “unidad virtual” del dispositivo físico.
- ***BluetoothDevice***: Al igual que el *Adapter*, el “dispositivo Bluetooth” representa el dispositivo con el que se quiere conectar, disponiendo de la información como su nombre, dirección y estado de la conexión.
- ***BluetoothSocket***: Representa la conexión entre dos dispositivos. Es como un “cable inalámbrico”, a través del socket dos dispositivos pueden intercambiar cualquier flujo de datos de manera fiable y ordenada.

Con estas tres clases, la conexión se reduce a identificar el adaptador y el dispositivo, y a continuación crear un socket para permitir la comunicación entre ellos.

El proceso que se realizará para realizar la conexión Bluetooth con el sensor se muestra en la figura 15

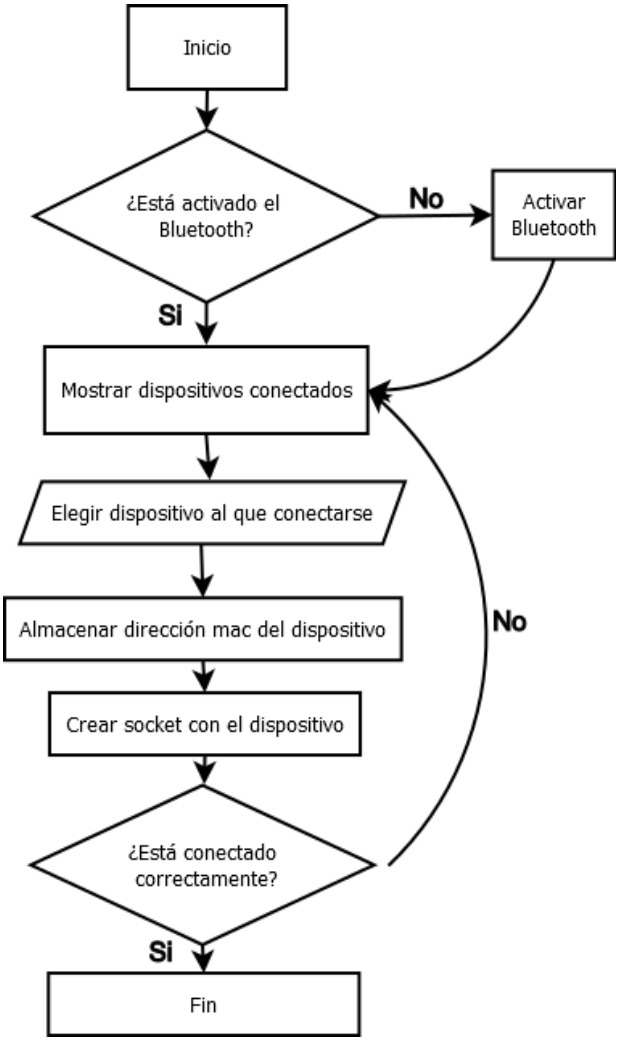

Figura 15: Conexión Bluetooth

El proceso a seguir sigue unos pasos concretos, en primer lugar prepara el Bluetooth, a continuación permite al usuario elegir a qué dispositivo conectarse y después realiza la conexión.

Se creó una aplicación simple utilizando ese proceso para permitir la conexión Bluetooth entre dos dispositivos.

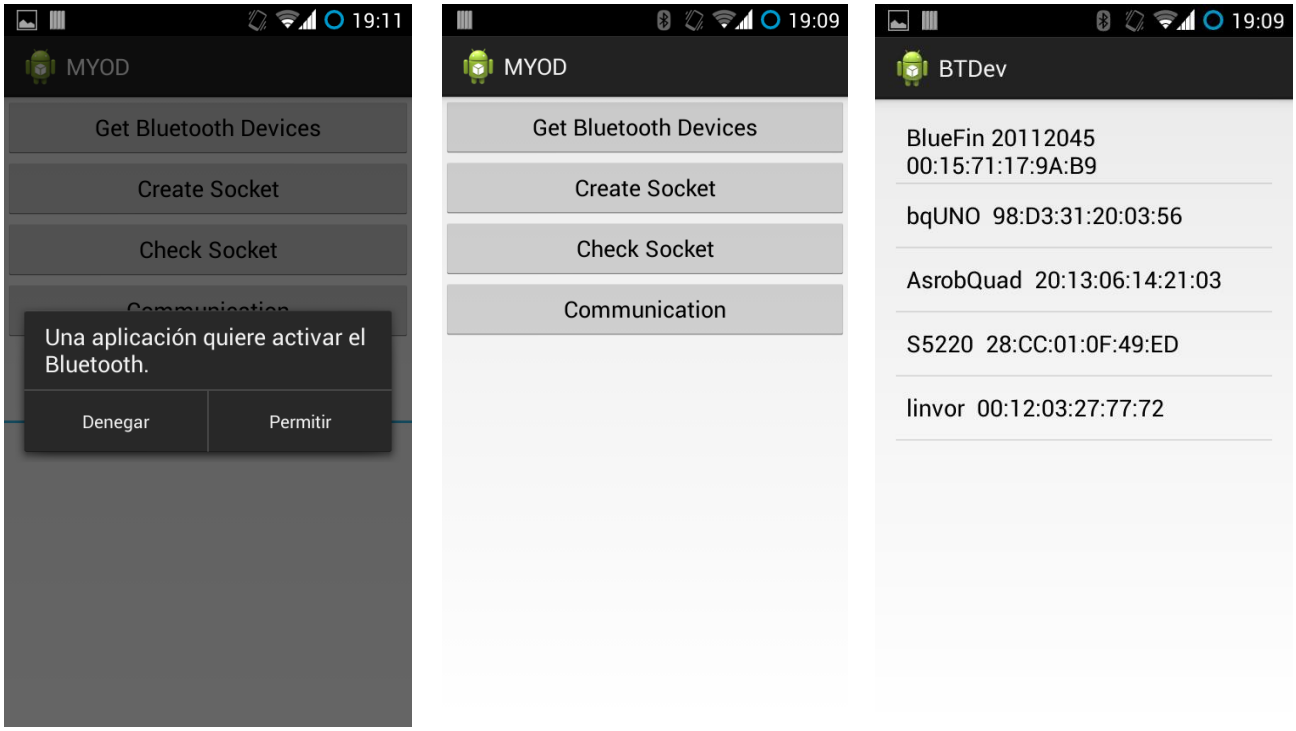

Figura 16: Pruebas Bluetooth

En la figura 16 se muestran capturas de pantalla del proceso para conectarse al dispositivo mediante Bluetooth. En primer lugar se activa el Bluetooth, para ello es necesario tener creada la instancia de la clase *BluetoothAdapter*, que representa la radio del dispositivo. Una vez activado se pueden mostrar los dispositivos conectados al móvil, que permite seleccionar uno de ellos, copiando su dirección MAC para poder usarla más adelante. Esta dirección MAC se almacena en una instancia de la clase *BluetoothDevice*, representando el dispositivo al que se quiere conectar. Una vez se dispone del adaptador y del dispositivo, únicamente es necesario crear el *BluetoothSocke*", estableciendo la comunicación entre ambos dispositivos.

Para mostrar por pantalla las reacciones del dispositivo y el estado de la conexión se decidió utilizar *Toast*, una clase de Android cuya función es mostrar por pantalla pequeñas reacciones a entradas sin alterar el funcionamiento del resto de la aplicación. Esto se muestra en la figura 17.

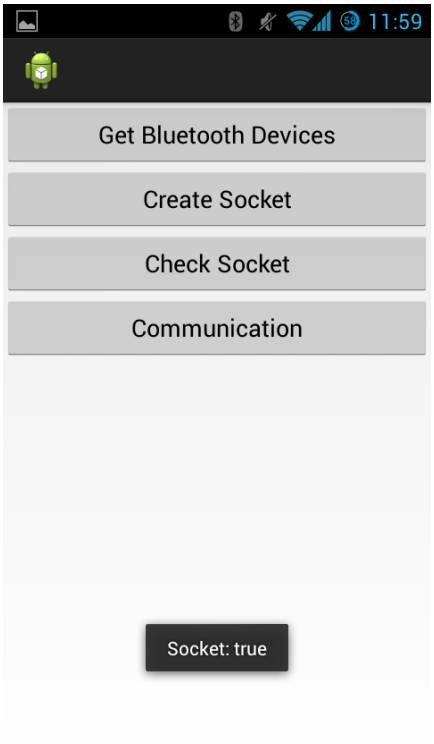

Figura 17: Toast en la aplicación Bluetooth

6.2.2 Pruebas del SDK del sensor.

Al recibir el sensor elegido, era necesario familiarizarse con el SDK (*Software Development Kit*) de éste para su correcta utilización. El SDK, provisto por el fabricante del sensor, se utiliza como una manera de comunicación con el sensor, para poder recibir la información recogida por él y tratarla correctamente. Este SDK contiene una serie de funciones necesarias para la correcta utilización del sensor.

En la figura 17 se muestra los componentes del SDK del sensor.

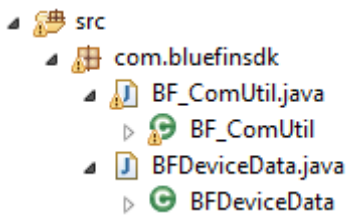

Figura 18: SDK Sensor BlueFin

En concreto, este SDK está compuesto por dos clases principales, *BF_ComUtil* y *BFDeviceData*.

BFDeviceData sería técnicamente la "unidad virtual" del sensor. En ella se encuentran las variables con la información del sensor (nombre, versión, estado de la batería, etc.) así como si el sensor tiene una imagen almacenada o no, y el lugar donde se almacena la imagen.

BF_ComUtil contiene todas las funciones necesarias para la comunicación con el sensor. Las más destacadas son:

- ***isIsInit***: permite comprobar que el dispositivo esté encendido y conectado.
- **BF_*SetSessionKey***: para poder recibir información del sensor, se necesita proveer una clave para que el sensor entre en modo seguro. Al entrar en modo seguro el sensor abre el resto de sus funciones para poder enviar datos correctamente.
- **BF_*Echo***: función simple que únicamente devuelve los datos que le ha sido enviados. Muy útil para comprobar su correcto funcionamiento.
- **BF_*GetFPImage***: comprueba que hay una imagen en el sensor y la envía a través del "socket".

Con ello se desarrolló una aplicación utilizando el SDK del sensor, que permitía comunicarse con el sensor recibiendo las imágenes capturadas por él. También se comprobaron las limitaciones del sensor y diversas características, como la duración de la batería, la capacidad de comunicación y la velocidad de captura.

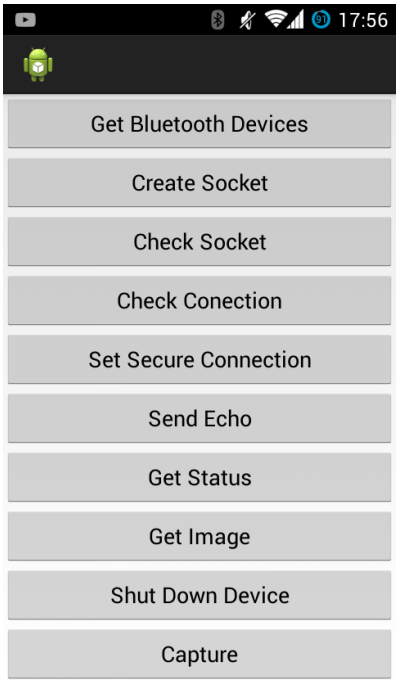

Figura 19: Aplicación prueba Bluetooth

Como se muestra en la figura 18, la aplicación consta de una actividad principal desde donde se puede acceder a todas las funciones que se quieren probar por medio de botones.

Las primeras funciones tienen relación con la conexión Bluetooth, explicadas en el apartado anterior 6.2.1. Estas funciones son necesarias para establecer la comunicación con el sensor.

Una vez se ha establecido la comunicación correctamente, se utilizará el SDK del sensor para recibir información de él.

En primer lugar, es necesario comprobar que el sensor está encendido y es capaz de enviar datos a través del socket previamente creado, para ello se utiliza la función isIsInit del SDK. Para utilizar esa función se pulsa *Check Connection* en la aplicación, que devolverá si la conexión es posible (*true*) o no (*false*) a través de los *Toast* explicados anteriormente.

Cuando se ha comprobado la conexión, es necesario establecer una comunicación segura con el sensor para que este pueda devolver información. Esto se realiza a través del botón *Set Secure Connection*, que utiliza la función *BF_SetSessionKey*. Esta función requiere de un código de seguridad que se almacena en la aplicación y en el sensor, si se provee el código correcto se establece la comunicación segura.

Get Status y *Send Echo* son utilizadas para comprobar que el sensor puede comunicarse, enviando datos como el nombre y estado del dispositivo y devolviendo los datos enviados por la aplicación.

La funcionalidad más importante de esta aplicación es el comprobar si el sensor puede realizar una captura de huella y enviársela a la aplicación. Activando el botón *Get Image*, se lanza una nueva actividad desde la que, utilizando la función *BF_GetFPImage* se puede pedir al sensor una imagen de huella. El sensor devolverá un *array* de bytes de una dimensión con la imagen, que la aplicación convertirá en una imagen para poder mostrarla por pantalla. Esto se ve en la figura 20.

Figura 20: Huella impresa por pantalla

Al realizar estas pruebas, se encontró un inconveniente con las características del sensor. Para realizar una captura de imagen, no es posible ordenar la captura directamente desde la aplicación, sino que el sensor detecta automáticamente si hay un dedo situado sobre él para realizar la captura. Una vez detecta el dedo, el sensor emite dos tonos, guardando la imagen en su memoria. La función *BF_GetFPImage* recibe la imagen de la memoria del sensor. Esto podría ser un problema, ya que ese sistema no es completamente fiable, permitiendo recibir imágenes en blanco si no se ha capturado una huella, y en ocasiones puede dar lugar a tener que esperar segundos hasta que se realiza la captura correctamente.

6.3 Implementación en BioApi

Una vez comprobado el correcto funcionamiento del sensor y el método de comunicación, el siguiente paso era implementar los métodos de comunicación con el sensor en el estándar de BioAPI.

Como se ha explicado en el apartado 3, la función de BioAPI consiste en generar una independencia de plataformas y dispositivos para los programadores y desarrolladores de servicios biométricos. La especificación e implementación de referencia de BioAPI es compatible con un amplio rango de aplicaciones biométricas y software, así como numerosas tecnologías biométricas distintas.

El primer paso para la implementación del sensor en la plataforma de BioApi consiste en la creación de una unidad de sensor.

El objetivo de esta unidad es que el programador no necesite preocuparse por las diferencias entre distintos sensores, ya que la función agrupa todas las funciones necesarias del SDK, realizando la captura correctamente.

En concreto, para este proyecto se necesitaba recibir la imagen de la huella dactilar y convertirla en un BIR, que es necesario para poder procesar la imagen más adelante.

Una vez creada la unidad de sensor es necesario implementarla en un BSP.

Un BSP agrupa las diversas unidades de BioApi, permitiendo un acceso a sus funciones de una manera homogénea.

Disponiendo de un BSP con unidades de archivo, procesado y comparación ya implementadas para sistemas biométricos de huella dactilar, se debía incluir la unidad de sensor desarrollada para el sensor de este trabajo.

Con el sistema completo, el BSP se incluye en un *framework* que funciona como paso intermedio entre la aplicación final y el BSP, dando acceso a todas las funciones de manera estándar.

En la figura 21 se puede ver la estructura del sistema.

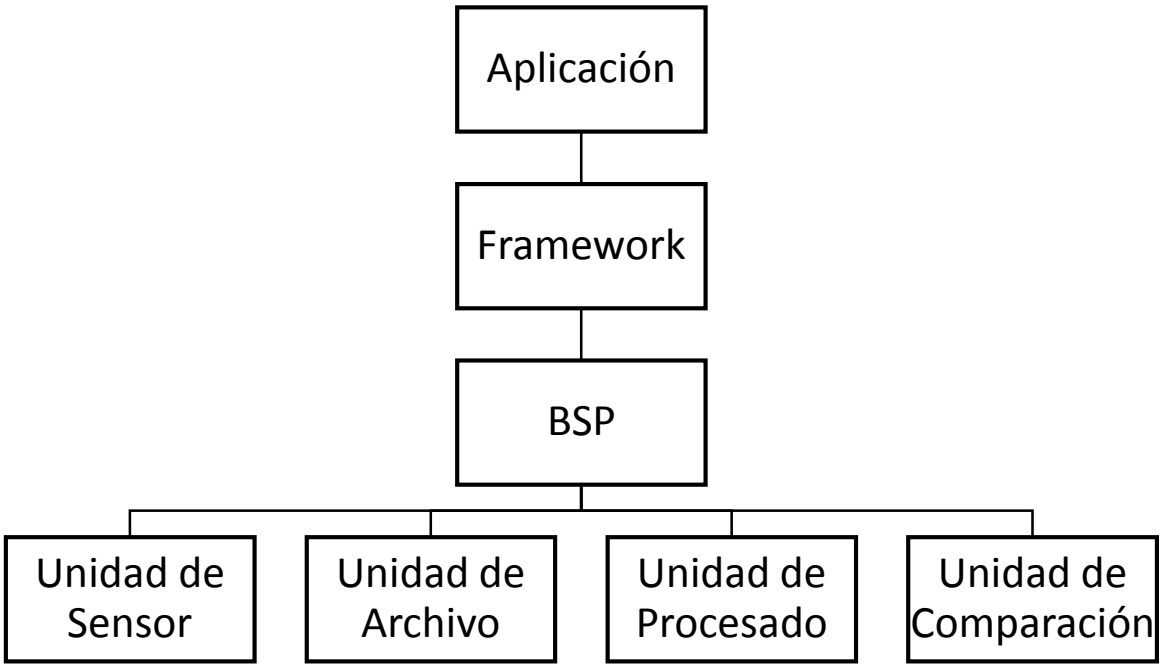

Figura 21: Esquema de la estructura de BioAPI

Durante este proceso se probaron todas las funciones por separado, para comprobar el correcto funcionamiento de éstas dentro de sus unidades.

A continuación se entra en detalle en el desarrollo de cada unidad.

6.3.1 Unidad de sensor

El primer paso para la implementación del sensor en la plataforma de BioApi consiste en la creación de una unidad de sensor.

En el sistema biométrico, la unidad de sensor es indispensable en el reclutamiento y la verificación, explicados en el apartado 2.3.

Una unidad de sensor consiste en agrupar todas las funciones necesarias para el correcto funcionamiento del sensor en una unidad privada incluida en el BSP.

Esta unidad tiene toda la información del sensor y de las imágenes capturadas por él, así como la información necesaria para conectarse a él, en este caso la dirección Bluetooth y las características del socket.

Como se ha explicado en el apartado 3.3.1, el componente más importante de la unidad es la función “capture”. El objetivo de esta función es obtener una imagen del

sensor. Por ello, en este caso es necesario introducir toda la parte de conexión Bluetooth, de comprobación de conexión y de obtención de la imagen.

De esta manera, se puede ordenar la función capture desde el BSP, sin tener que preocuparse por la conexión con el sensor, y obteniendo una imagen directamente.

El proceso que sigue la unidad de sensor es el siguiente:

En primer lugar, cuando se crea una instancia de la unidad se inicializa el sensor y su SDK. Una vez inicializado, se crea el socket de comunicación Bluetooth como se ha visto en el apartado 6.2.1. Para ello anteriormente se debe disponer de la dirección MAC del sensor. Una vez conectado, se comprueba que la conexión es correcta. Esto se realizará preferiblemente al iniciar la aplicación, ya que solo es necesario hacerlo una vez.

Para capturar una huella, se llamará al método *capture*. Este método utilizará la función *BF_GetFPImage* del SDK del sensor como se ha explicado en el apartado 6.2.2, almacenando lo recibido en un *array* de bytes. Ese *array* de bytes será entonces incluido en un BIR, que preparará la imagen para que pueda ser procesada. Todo el proceso de captura tiene un límite de tiempo de 3000 milisegundos para evitar que se bloquee la función en el caso de que surja algún problema. De esta manera, la función “capture” devolverá un BIR con la imagen capturada y preparada para ser procesada o almacenada.

La función *capture* será llamada por los métodos *enrol* y *verify* dentro del BSP, que serán explicados más adelante.

En el apartado 6.4.2 se explicará el funcionamiento de este proceso en la aplicación final.

6.3.2 Unidad de procesado

Unidad en la que se encuentran los algoritmos que procesan la imagen de huella dactilar y extraen la información biométrica de la misma.

Como se ha explicado en el apartado 2.4 sobre la huella dactilar, ésta dispone de ciertas características como el patrón y las minucias. Para facilitar la comparación de la huella se puede procesar para obtener información de esas características.

El método fundamental de esta unidad es el método *process*. Como se ha explicado en el apartado 3.3.1 recibirá un BIR capturado (mediante el método capture) con la imagen convertida a un *array* de bytes. Una vez recibido el BIR, se extraerá el *array* para procesarlo.

Una vez se dispone del *array* de bytes, se ejecutan los algoritmos proporcionados. En concreto, para el procesado de las huellas dactilares se utilizará el algoritmo Mindtct. [18] [19]

El algoritmo Mindtct procesa un archivo que contenga una imagen de huella dactilar en escala de grises, detectando las minucias de la huella y almacenándolas.

En la figura 22 se observan dos tipos de minucias, una bifurcación de cresta y un final de cresta. Normalmente en una huella dactilar existen alrededor de 100 minucias. El algoritmo detecta las minucias y almacena su posición y la orientación de las crestas.

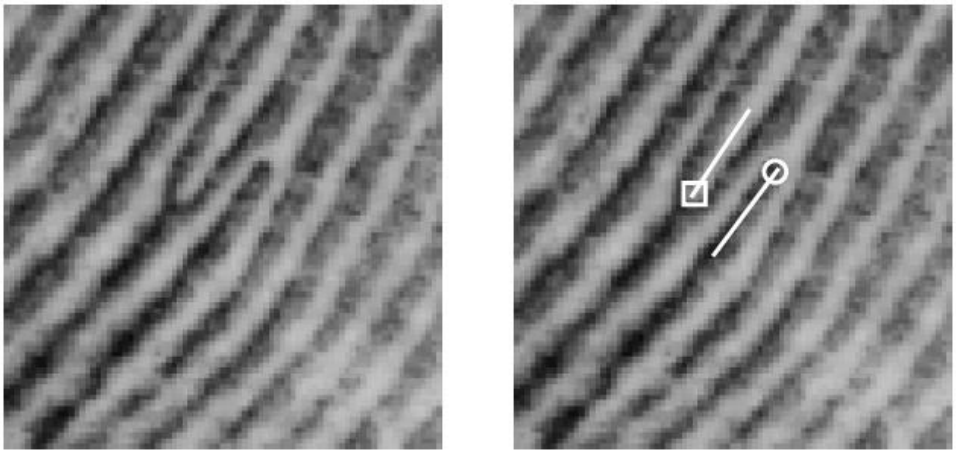

Figura 22: Minucias [18]

Para detectar la posición de las minucias, el algoritmo filtrará la imagen hasta reducirla a valores binarios (negro o blanco), tal como se ve en la figura 23. De esta manera, cada pixel solo puede tener dos valores, simplificando la identificación de las minucias a los casos que se observan en la figura 24. Los resultados no son definitivos, ya que la imagen puede ser de poca calidad o mal tomada, por lo que se eliminan candidatos como minucias que se sobreponen, que son muy anchas o muy estrechas.

Figura 23: Binarización de huella [18]

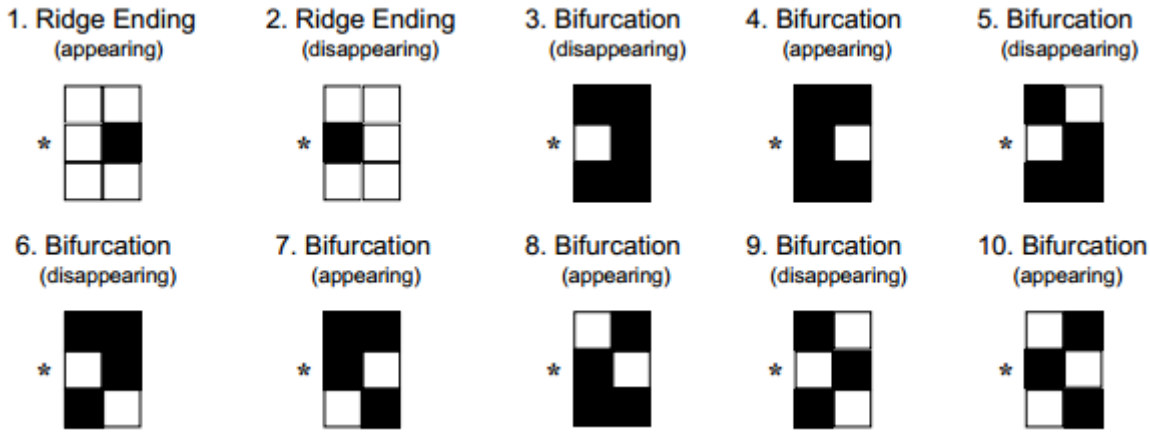

Figura 24: Detección de minucias [18]

Una vez detectada la posición y orientación de las minucias, es necesario almacenarla. Para ello, el algoritmo divide los pixeles de la imagen por coordenadas y representa la orientación en grados con respecto a la horizontal apuntando a la derecha y aumentando al contrario de las agujas del reloj.

Una vez procesada la huella, se creará un *array* con las características biométricas de la imagen, ya preparadas para su comparación. Con este *array* se creará un nuevo BIR y se almacenará.

Todo el proceso también será limitado en tiempo por si surgen problemas.

6.3.3 Unidad de archivo

Esta unidad se encarga de almacenar los BIR capturados. Para ello, utiliza una base de datos SQL.

SQL, lenguaje de consulta estructurado, es un lenguaje de acceso a bases de datos relacionales. SQL permite crear y realizar operaciones en las bases de datos de forma sencilla.

Al inicializar la unidad de archivo, se crea automáticamente una base de datos.

El funcionamiento de esta unidad es bastante sencillo: por medio del método *storeBIR* se puede almacenar un BIR directamente a una base de datos previamente creada. A ese BIR se le asigna automáticamente un UUID aleatorio, que servirá para poder identificar el BIR al sacarlo de la base de datos.

Para sacar un BIR, se puede usar la función *getSingleBIR*, que recibe un UUID y devuelve el BIR asociado.

También es posible acceder a toda la base de datos o a una serie de datos en concreto mediante el método *identifyPopulation*. Este método permite devolver un grupo de UUID almacenados en la base de datos y será usado en la unidad de comparación para comparar un BIR con la base de datos.

Para poder tratar con la base de datos es necesario, al ser del tipo SQL, abrir la base de datos antes de realizar operaciones en ella y cerrarla al terminar. Esto evita problemas de pérdida de datos o datos erróneos.

Un inconveniente encontrado en el funcionamiento de la unidad de archivo es el que no permite almacenar datos que no sean BIR o sus respectivos UUID. Esto está pensado de tal manera que BioAPI se centre únicamente en información biométrica.

Como se explica más adelante, para poder almacenar más información (nombre de la huella y fecha) se ha necesitado crear una nueva base de datos externa a BioAPI que almacena esa información extra en paralelo con la base de datos principal.

6.3.4 Unidad de comparación

Unidad en la que se encuentran los algoritmos necesarios para comparar los *arrays* de características biométricas.

Para realizar la comparación se utilizará el algoritmo Bozorth3 [18]. Este algoritmo genera un valor de correspondencia (FMR) entre dos archivos de minucias de huella dactilar. El algoritmo compara la posición y orientación de las minucias, que se han obtenido previamente al procesar la huella.

El algoritmo sigue el siguiente proceso:

- En primer lugar realiza mediciones de la distancia y el ángulo de cada minucia a las demás de la misma huella y las almacena en tablas de comparación.
- A continuación, se comparan las tablas de comparación de dos huellas buscando entradas compatibles. Las entradas compatibles se comprueban en base a tolerancias. De ese modo, si los valores de una minucia en una huella son similares a los de otra minucia en otra huella ambos son compatibles.
- Por último se agrupan las entradas compatibles en “racimos” y se le otorga un valor de comparación. Cuanto más grandes sean los “racimos” compatibles entre dos huellas, mayor será el resultado, y existirá mayor posibilidad de que la huella sea de la misma persona.

Este algoritmo está implementado en BioAPI en una unidad de comparación.

El método más relevante dentro de esta unidad es el método *verify*. Este método recibirá dos BIR ya procesados y devolverá un valor de relación entre los dos BIR.

La unidad también dispone de un método llamado *identify*. Este método toma como argumentos un BIR que se quiere comparar, un vector de BIR y un valor para el FMR. De esta manera, con este método se puede comparar un único BIR con una serie de BIR distintos, devolviendo los que superen el valor del FMR provisto.

En éste trabajo se utiliza el método *identify* para realizar la identificación de un usuario comparándola con los usuarios almacenados en la base de datos.

Esta unidad será utilizada en el método *verify* en el BSP.

6.3.5 *Biometric Service Provider* (BSP)

Como se explica en el apartado 3.3, un BSP agrupa las diversas unidades de BioAPI, permitiendo un acceso a sus funciones de una manera homogénea. El BSP evita tener que especificar que sensor o que unidad se quiere utilizar, simplificando el trabajo realizado por el usuario. Así mismo, el BSP permite mayor versatilidad al unir diversas funciones de las unidades.

En el trabajo se utiliza el BSP para acceder a las funciones incluidas dentro de las unidades, tales como *storeBIR* de la unidad de archivo o “capture” de la unidad de sensor. Además, el BSP tiene funciones propias muy importantes para el desarrollo del trabajo:

- ***Enrol*:**

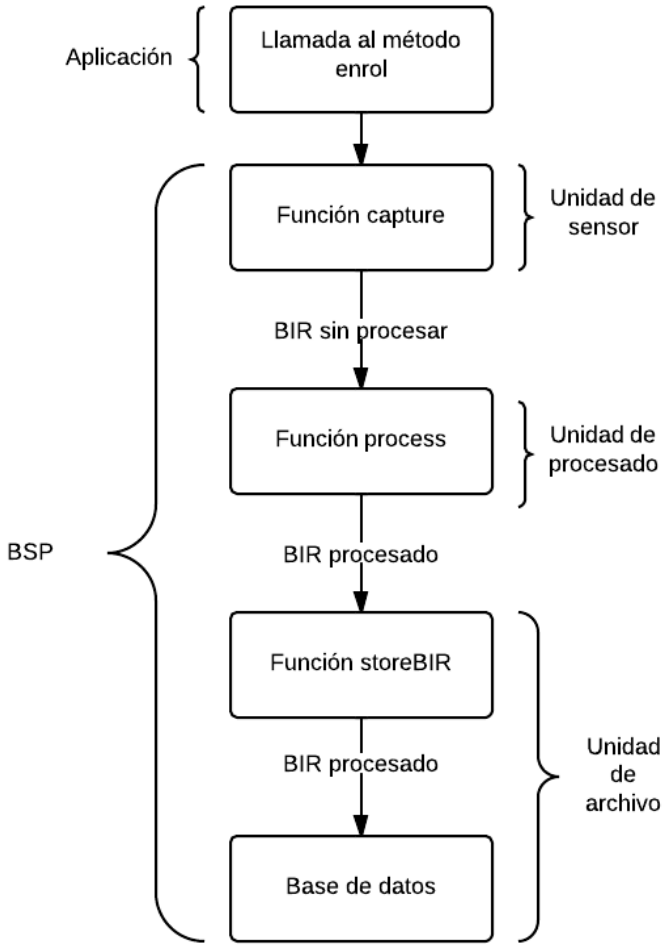

Figura 25: Método *Enrol*

Enrol se encarga de gestionar completamente el registro de un nuevo usuario en el sistema. Engloba todas las llamadas a funciones necesarias para completar el proceso. En este caso en particular, *enrol* se encargará en primer lugar de realizar una captura. Una vez dispone del BIR capturado, lo procesará por medio de la función *process*, generando otro BIR con la imagen procesada, y lo almacenará en la base de datos. Como se ve, agrupa funciones de las unidades de sensor, procesado y archivo.

- ***Verify*:**

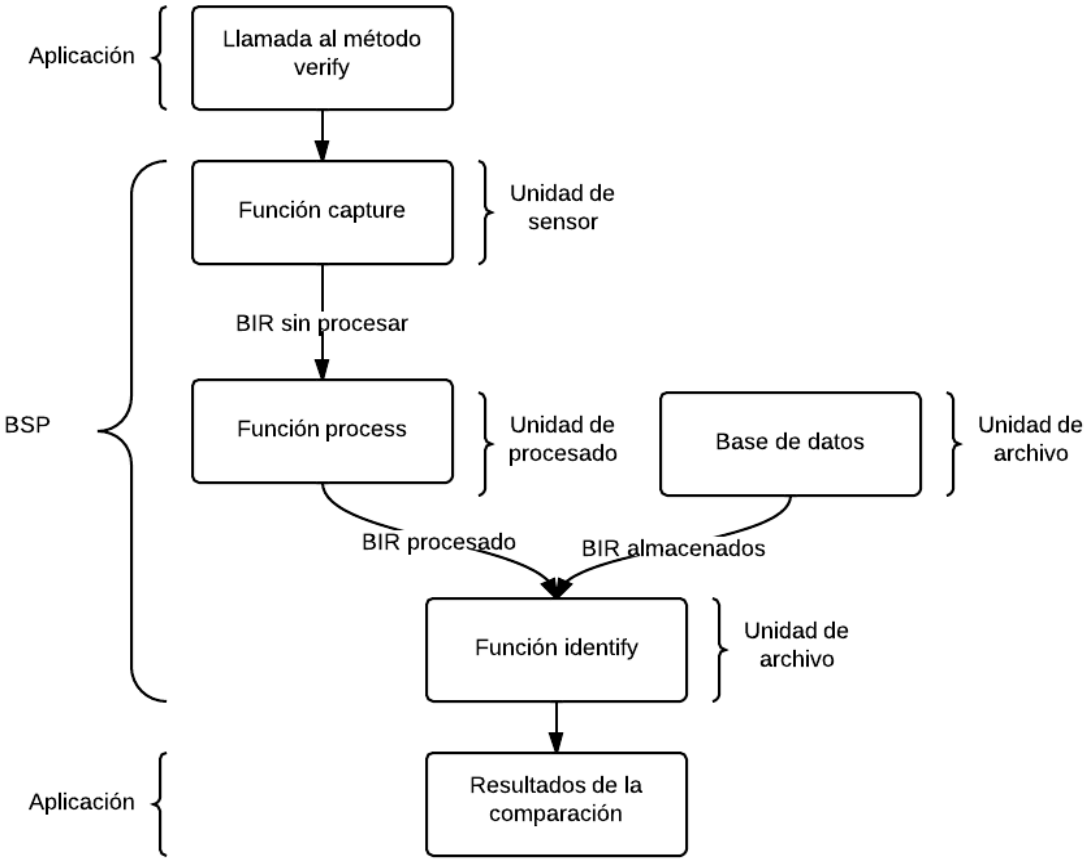

Figura 26: Método Verify

Verify se encarga de realizar las llamadas necesarias cuando se quiere realizar una verificación. Al llamarlo, realizará una captura, la procesará y la comparará con uno o varios BIR almacenados en la base de datos, usando las funciones *verify* o *identify* de la unidad de comparación. Como resultado devolverá el FMR de las comparaciones.

En definitiva, el BSP agrupa todas las unidades, provee de distintas interacciones entre ellas y permite el acceso desde niveles superiores del sistema biométrico.

6.3.6 Framework

Como se comentó en el apartado 3.2, el *framework* es el elemento encargado de gestionar la comunicación entre la aplicación y los BSP, así como de controlar todos los elementos conectados a él, de esta manera se puede comprobar la disponibilidad de los diversos BSP por la aplicación, ya que ésta no tiene acceso directo a ellos.

En este caso, al inicializar el *framework* se "instalarán" el BSP y sus unidades, permitiendo decidir qué unidades serán utilizadas y cuáles no.

De esta manera se puede saber si existen problemas en alguno de los módulos y podría permitir ampliar el sistema añadiendo otro BSP u otras unidades.

Como se ha mencionado, no es necesario la utilización de un *framework*, ya que sólo se trabajará con un BSP, siendo el f*ramework* una manera de simplificar la utilización de múltiples BSP. Sin embargo se ha decidido utilizarlo para mostrar un entorno BioAPI completo y permitir posibles ampliaciones futuras.

6.4 Creación de aplicación

Una vez todos los elementos están funcionando correctamente por separado, se crea una aplicación de ejemplo para mostrar el comportamiento de los elementos en conjunto.

La aplicación dispone de diversas funciones:

- Conexión Bluetooth con el sensor: permite elegir el sensor de la lista de dispositivos vinculados con el móvil Android
- Captura: Con la aplicación se puede capturar una imagen de una huella dactilar provista por el sensor. Una vez capturada la huella se procesará y se almacenará el BIR en una base de datos.
- Verificación: Haciendo otra captura de huella dactilar, se puede comparar con los BIR almacenados, devolviendo las correspondencias y el resultado de la comparación con los BIR almacenados que hayan recibido mayor puntuación.
- Base de datos: Dispone de una manera de comprobar las entradas almacenadas en la base de datos, viendo información sobre ellas.

A continuación se mostrará el funcionamiento de la aplicación completa, para luego profundizar en el desarrollo de cada apartado.

6.4.1 Funcionamiento de la aplicación

El funcionamiento básico de la aplicación se muestra en la figura 27:

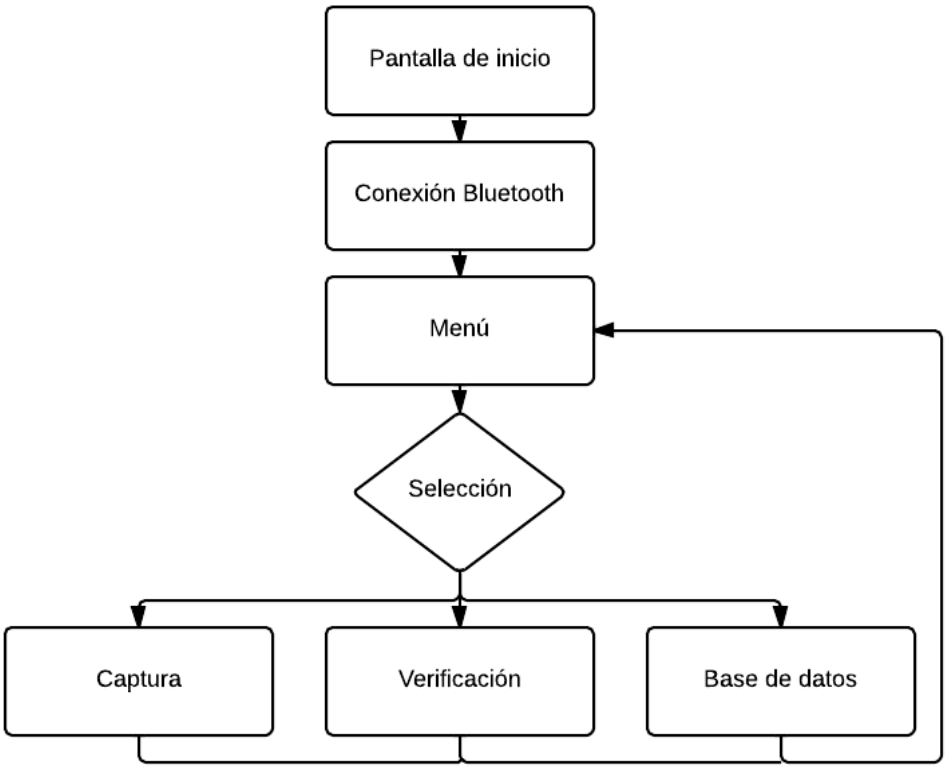

Figura 27: Diagrama general de la aplicación

Al iniciarse, pide al usuario que active la conexión Bluetooth del dispositivo, para seguidamente mostrar una lista de los dispositivos con conexión Bluetooth disponibles:

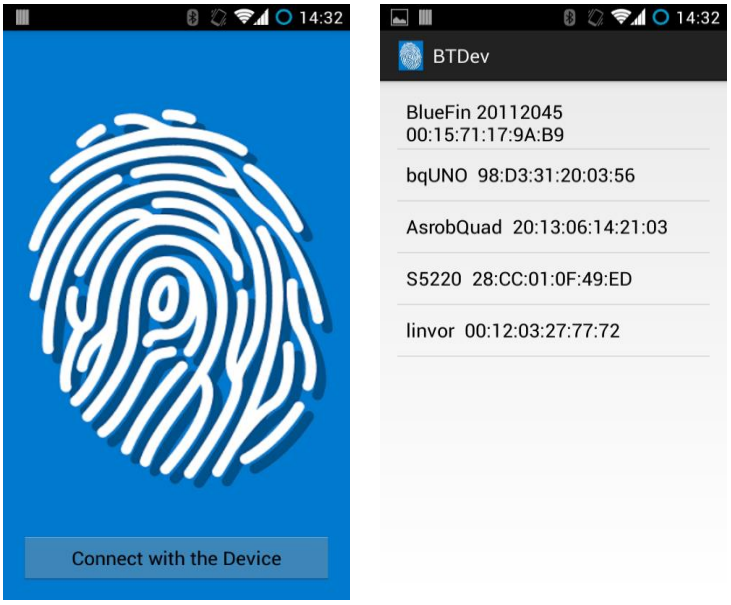

Figura 28: Conexión Bluetooth

El usuario podrá elegir el dispositivo con el que quiera comunicarse, en este caso el sensor BlueFin.

Una vez establecida la conexión, se muestra una pantalla con la función de menú principal, desde la que se podrá acceder a las distintas operaciones de la aplicación:

Figura 29: Menú

En la pantalla de captura ("*Capture*"), se permitirá pedir una imagen de la huella dactilar al usuario, comunicándose con el sensor. En el caso de que el sensor disponga de una huella disponible, ésta aparecerá en la pantalla y le dará la opción al usuario de nombrarla para su procesado y almacenado:

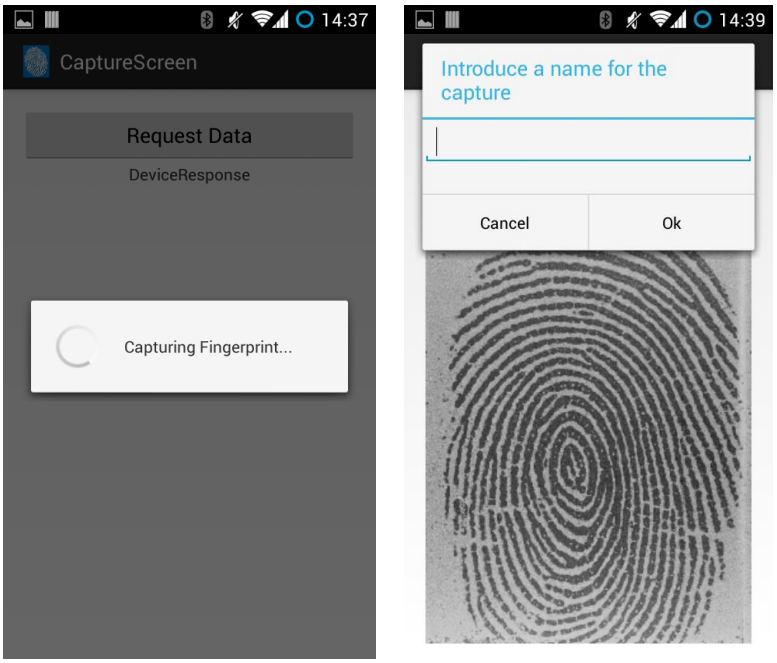

Figura 30: Captura de huella

En la pantalla de verificación ("*Verify*") se capturará una huella, pero en este caso la huella será procesada y comparada con la información almacenada en la base de datos, devolviendo, en el caso de su existencia, resultados de comparación con las entradas de la base de datos más similares:

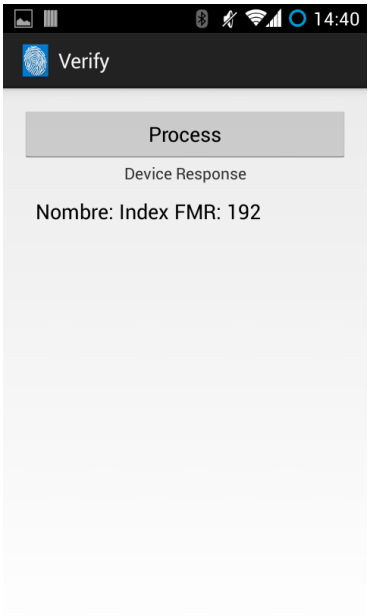

Figura 31: Resultados de verificación

En la pantalla base de datos ("*Database*") se muestran las entradas almacenadas anteriormente, con la fecha de la captura y el nombre. Se puede acceder a una pantalla individual de cada entrada en la que se muestran sus datos:

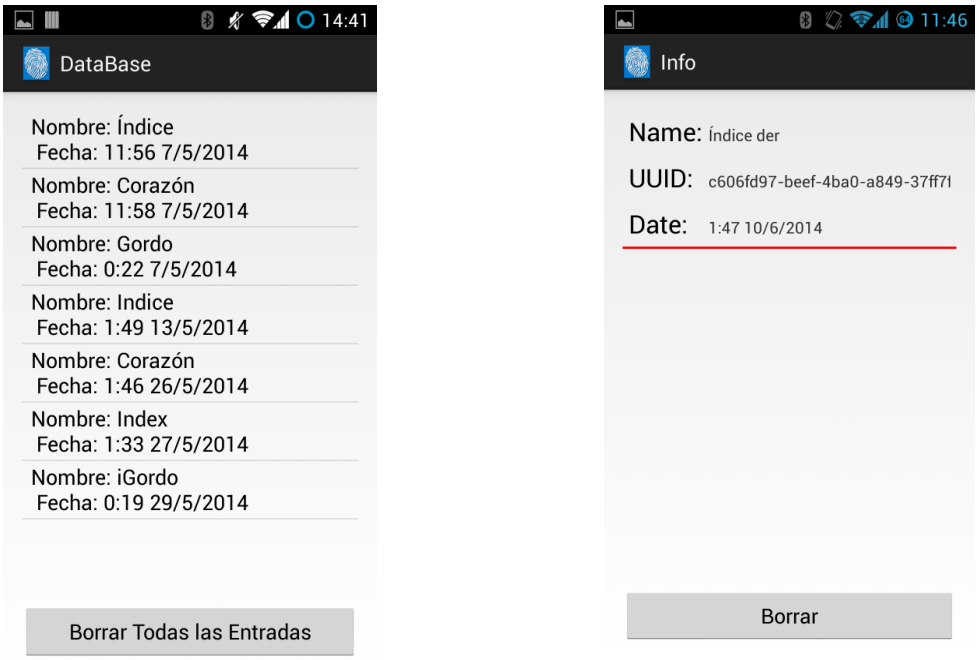

Figura 32: Base de datos

A continuación se entrará en detalle en cada uno de los estados de la aplicación, explicando las funciones más importantes y su proceso de ejecución.

6.4.2 Descripción de la aplicación

Para facilitar el desarrollo de la aplicación, está separada en 4 proyectos: librerías de BioAPI, *Framework*, BSP (junto al SDK del sensor y clases de apoyo) y la aplicación principal.

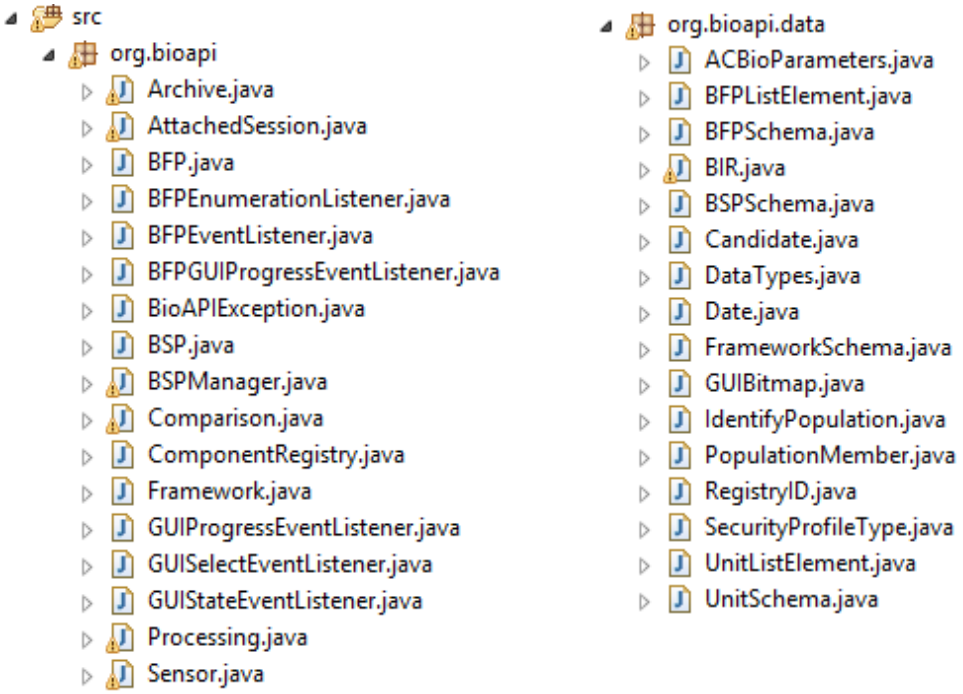

Figura 33: Librería de BioAPI

La librería de BioAPI (figura 33) contiene todas las clases y funciones que se utilizarán dentro del *framework* y del BSP.

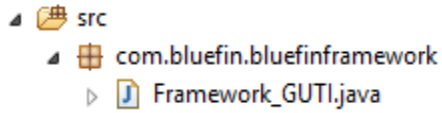

Figura 34: *Framework*

El *framework* (figura 34), como se ha explicado en el apartado 3.2, permite a la aplicación comunicarse con el BSP y sus unidades. En este proyecto se encuentran todas las funciones necesarias para la correcta inicialización del sistema BioAPI, que permitirán cargar el BSP y sus unidades y gestionarlos.

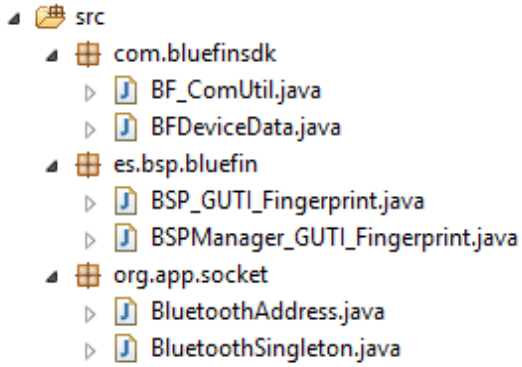

Figura 35: BSP, SDK del sensor y apoyo Bluetooth

En este proyecto (Figura 35) se agrupan el BSP, el SDK del sensor y clases de apoyo para el funcionamiento de la conexión Bluetooth. Como se ha explicado en el apartado 3.2, la aplicación no tiene acceso directo al BSP, sino que utiliza el *framework* para comunicarse con él. El BSP agrupa todas las unidades del sistema BioAPI y dispone de los métodos necesarios para las distintas funcionalidades del sistema biométrico.

El SDK del sensor contiene todas las funciones utilizadas para la comunicación con el sensor (apartado 6.2). Estas funciones serán utilizadas por la unidad de sensor del BSP.

Las clases de apoyo para la conexión Bluetooth serán utilizadas también por la unidad de sensor para establecer la conexión con él.

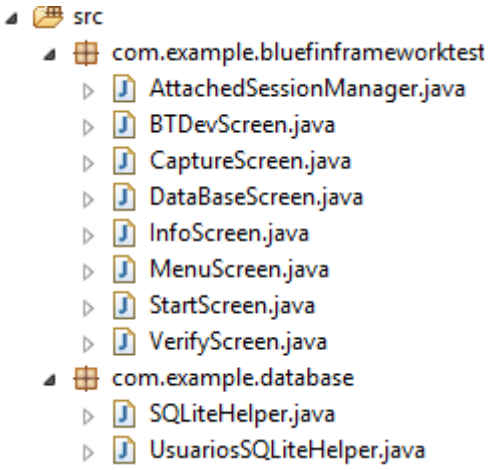

Figura 36: Aplicación

En la figura 36 se muestra la estructura de clases de la aplicación. En ella se encuentran todas las clases desarrolladas para las distintas actividades de las que consta la aplicación, así como las clases de apoyo para la base de datos y la gestión del *framework* BioAPI.

6.4.2.1 Inicialización de la aplicación y conexión con el dispositivo Bluetooth

Cuando se inicia la aplicación se lanza la primera actividad. La función de esta actividad es introducir la aplicación y comprobar que el dispositivo en el que se está ejecutando tiene la capacidad de poder llevar a cabo los procesos necesarios.

Nada más inicializarse la actividad, se comprobará si el dispositivo dispone de Bluetooth. Si el dispositivo tiene Bluetooth, se comprobará si está conectado, lanzando un *Dialog* en el caso de que no esté conectado. Un *Dialog* es una pequeña ventana que le pide al usuario que tome una decisión, en este caso, si desea o no activar el Bluetooth del dispositivo, lo que se puede ver en la figura 37. En el caso que no se active el Bluetooth la aplicación acabará, ya que es necesario para la correcta inicialización del sistema.

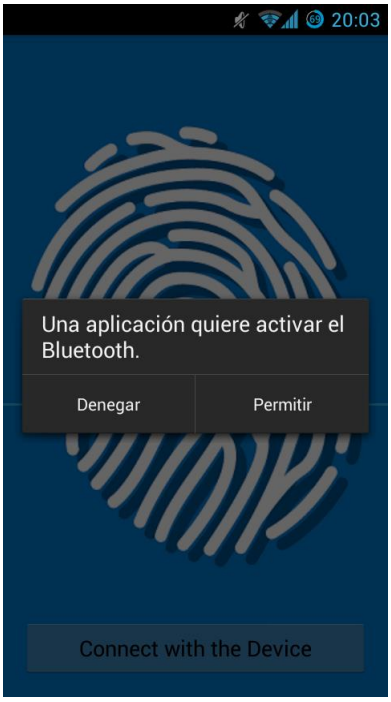

Figura 37: Dialog de conexión Bluetooth

Una vez activado el Bluetooth, el usuario podrá pulsar el botón *Connect with the Device*, que lanzará una nueva actividad para elegir un dispositivo al que conectarse.

Esta nueva actividad seguirá el proceso utilizado en el apartado 6.2.2. Utilizando las funciones provistas por el API de Android para conexión Bluetooth, la aplicación accederá a la lista de dispositivos Bluetooth disponibles, mostrándolos en una lista por pantalla. De este modo, el usuario podrá elegir el dispositivo al que desea conectarse. Una vez elegido el dispositivo, se obtendrá la dirección MAC de éste, y se almacenará en una clase auxiliar, llamada *BluetoothAddress*, para que el resto de actividades puedan acceder a la dirección sin problemas.

Una vez elegido el dispositivo y almacenada su dirección MAC, se podrá inicializar el sistema completo.

6.4.2.2 Inicialización de la actividad principal y BioAPI

Una vez realizados los procedimientos previos para la elección del dispositivo Bluetooth, se procede a la inicialización de la actividad principal del sistema.

Esta actividad funcionará como el menú de la aplicación, por lo que será visitada numerosas veces en la misma vida de la aplicación sin ser destruida.

Después de la creación de la actividad, siendo el primer acceso a ésta, se inicializará BioAPI, así como todos los elementos necesarios para el correcto funcionamiento de

la aplicación. Este proceso requiere muchos recursos, no es buena idea que lo realice el hilo principal de la aplicación, por ello se ha creado un hilo secundario para realizarlo. Si el usuario interactuase con la aplicación antes de que esté correctamente inicializada, daría lugar a error terminando la aplicación, ya que no podría acceder a los sistemas necesarios para su funcionamiento. Para evitar que el usuario pueda interactuar, se mostrará un icono de progreso en una ventana, impidiendo el contacto con los elementos de la aplicación, tal como se muestra en la figura 38.

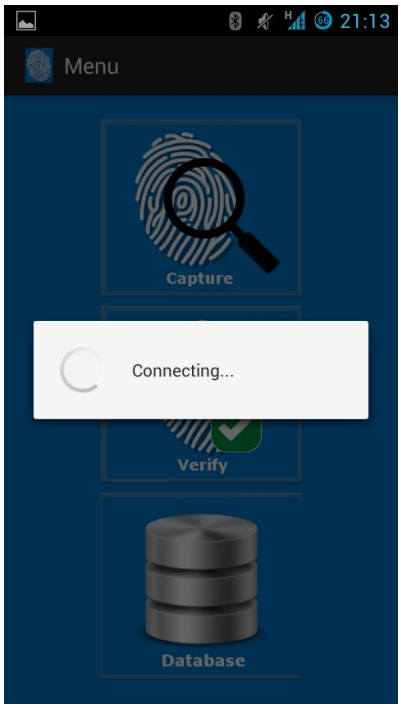

Figura 38: Ventana de progreso

El proceso de inicialización sigue el siguiente orden: en primer lugar se inicia el f*ramework* de BioAPI, seguido de todos los BSP disponibles en el *framework*, en este caso solo un BSP. Una vez cargado el BSP, se iniciarán las distintas unidades dentro del BSP, en este caso 4 unidades, sensor, archivo, procesado y comparación.

Todos los pasos de la inicialización disponen de sistemas para gestionar las excepciones lanzadas, para, en el caso de que exista un error, poder identificarlo fácilmente.

El paso más crucial de este proceso es la inicialización del sensor. Esto es debido a que, como se ha explicado en el apartado 6.4.1, la unidad de sensor se ocupa de la comunicación con el sensor de huella dactilar. En la inicialización, la unidad recibirá la dirección MAC del dispositivo elegido para la conexión, y tratará de conectarse a él por medio de las funciones del SDK. Todo ello puede dar lugar a diversos problemas. En primer lugar, la dirección MAC debe ser de un sensor de huella dactilar del modelo

propietario del SDK, no puede ser otro modelo ni otro dispositivo. El dispositivo debe estar encendido para realizar la conexión y no puede estar conectado a otro dispositivo u a otra instancia de la misma aplicación. En el caso de que el dispositivo sea equivocado o que pase el periodo de tiempo disponible para la conexión, la aplicación se reiniciará.

Para poder acceder a todos los métodos del BSP desde cualquier clase de la aplicación, se utiliza una clase tipo *singleton*, cuya intención es garantizar que una clase sólo tenga una instancia, que será accesible por cualquier clase. De esta manera, la inicialización sólo se realiza una vez, manteniendo sus valores durante la vida de la aplicación, así como los datos de la conexión con el sensor.

Una vez terminada la inicialización del sistema con todos los elementos correctos, la ventana de progreso desaparecerá y se mostrará la pantalla del menú principal.

Figura 39: Menú de la aplicación

Como se ha mencionado previamente, esta pantalla permite el acceso a las distintas operaciones de la aplicación. Al pulsar uno de los botones se lanzará la actividad correspondiente.

6.4.2.3 Capture

El objetivo de esta actividad es el realizar una captura de huella del usuario, para seguidamente almacenarla. Esto lo realizará por medio del método *enrol* explicado en los apartados 3.3 y 6.4.5.

El proceso de captura es el siguiente:

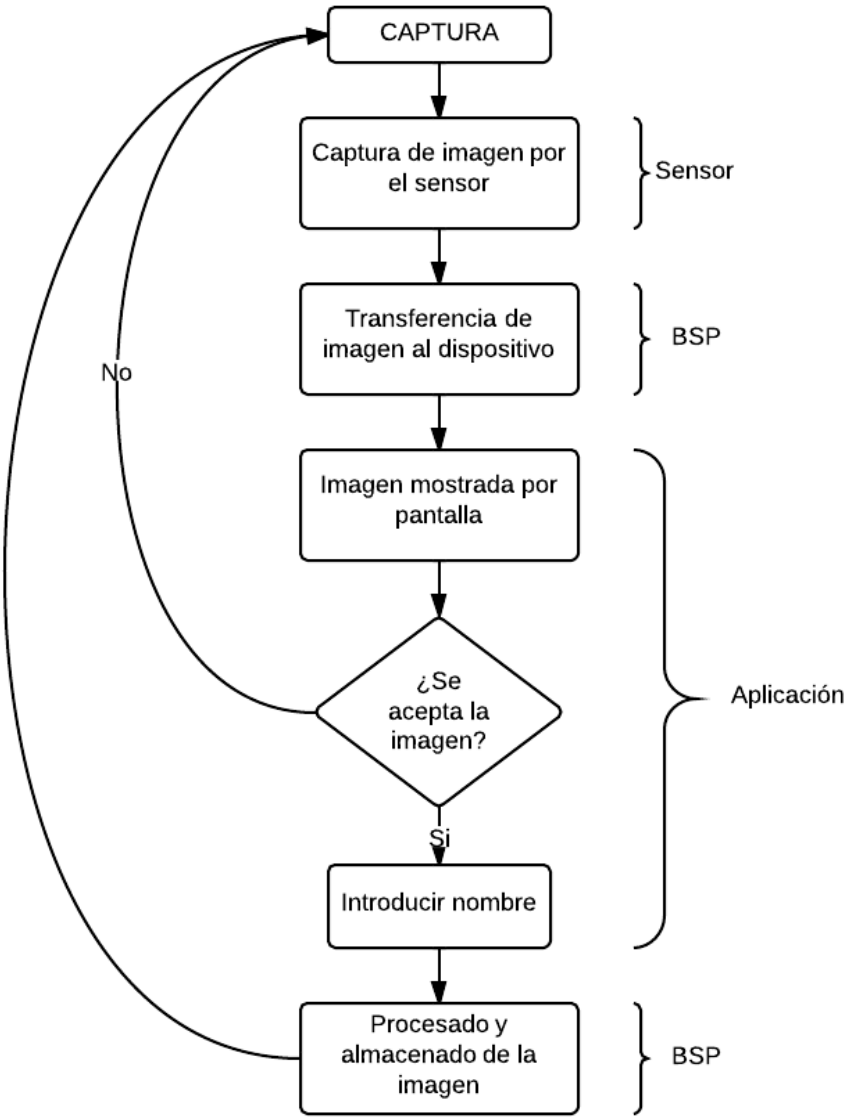

Figura 40: Proceso de captura

En primer lugar, el usuario deberá poner el dedo en el sensor de huella hasta que suenen dos tonos. Una vez suenen los tonos significa que el sensor dispone de la imagen de la huella en su memoria interna. Para acceder a ella, el usuario debe pulsar el botón *Request Data* comenzando el proceso de captura.

Al igual que el proceso de inicialización, este proceso es muy exigente, así que se ha creado un hilo para realizar la captura, oculto también tras una ventana de progreso.

Para realizar el *enrol* de la huella, en primer lugar se realizará una captura provisional mediante la función “capture”. Esta función almacenará la imagen en un BIR y a continuación la mostrará por pantalla. Para mostrar la imagen por pantalla se ha utilizado la interfaz de java *EventListener* que incluye BioAPI. Gracias a esta interfaz, se puede reaccionar a eventos que ocurran en el interior de BioAPI desde la aplicación. De esta manera, se ha utilizado para que cuando la función “capture” que se encuentra dentro de la unidad de sensor del BSP, a la que la aplicación no tiene acceso directo, llegue al punto en el que ha recibido la imagen de la huella, esta imagen pueda ser utilizada por la aplicación, sin necesidad de esperar a que terminen el resto de procesos o de obtenerla del BIR. Cuando el evento salta, la imagen aparece en la pantalla.

Una vez la imagen aparece por pantalla, a su vez aparece un *dialog*. Este *dialog* permitirá al usuario decidir si quiere almacenar la huella capturada e introducir un nombre.

En el caso de que no se desee almacenar la huella, ésta se borrará y se permitirá capturar una nueva. Si la respuesta es afirmativa, el proceso de *enrol* proseguirá, procesando el BIR capturado mediante la función *process* explicada en el apartado 6.4.2, e introduciendo el BIR en la base de datos de huellas de BioAPI como se ha explicado en el apartado 6.4.3. El nombre introducido para la huella no se puede introducir en la base de datos destinada a BIR, ya que en BioAPI se pretende separar la información biométrica del resto. Para almacenar el nombre se ha creado otra base de datos que asigna un nombre a cada UUID de cada BIR.

Una vez termina el proceso, se puede realizar una nueva captura, o se puede regresar al menú principal.

6.4.2.4 Verify

Esta actividad realizará la comparación de muestras biométricas nuevas con las almacenadas en la base de datos. Para ello, utilizará el método *verify* incluido en el BSP.

Proceso de verificación:

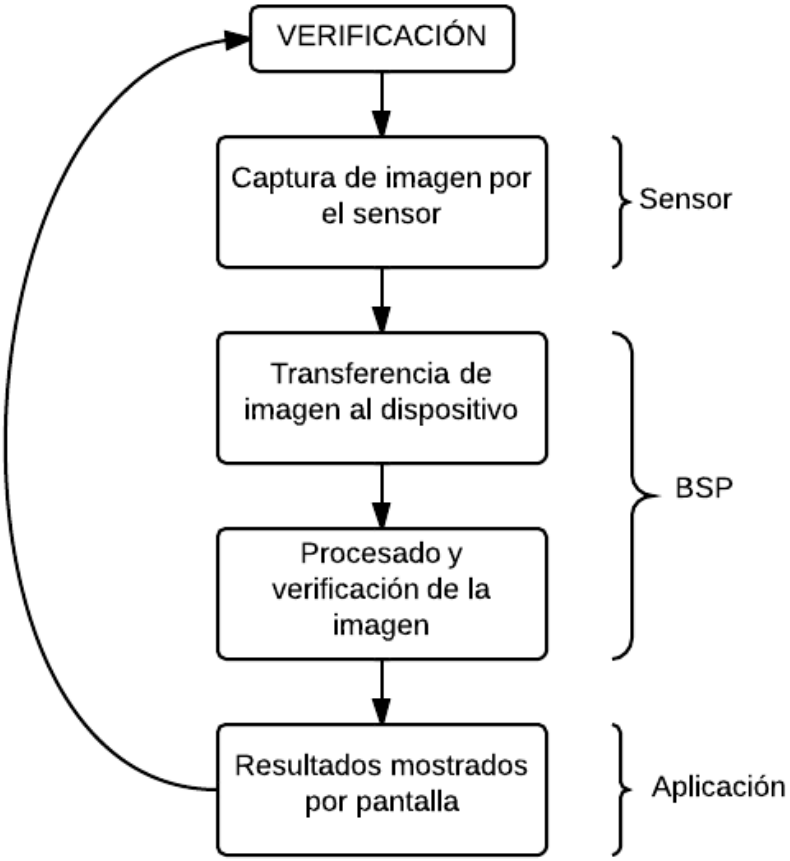

Figura 41: Proceso de verificación

La actividad tiene una estructura similar a la actividad *Capture* en el apartado anterior. La primera parte es igual, ya que se necesita capturar una imagen de huella dactilar, el usuario deberá poner el dedo en el sensor y a continuación pulsar *Request Data*. Una vez la aplicación dispone de la huella, no necesita mostrarla por pantalla, por lo que directamente la procesa. Con la imagen procesada, la aplicación realizará la función *identify* explicada en el apartado 6.3.4, utilizando como población para la identificación toda la base de datos almacenada. De esta manera la huella capturada se compara con todas las almacenadas. Este proceso se realiza en un hilo alternativo, ya que es muy exigente para el dispositivo.

Una vez se dispone de los resultados de la comparación, se filtran todos los que dispongan de un FMR menor de 60, y, en el caso de que existan, se muestran los 2 mejores resultados por pantalla con sus nombres y la fecha de adquisición.

Se puede acceder a una pantalla individual con toda la información de cada resultado.

6.4.2.5 Database

Para el funcionamiento del sistema es necesaria la utilización de diversas bases de datos.

La unidad de archivo incluida en el BSP se encarga del almacenamiento y la gestión de los BIR, y es controlada por el BSP.

Para la utilización de las bases de datos creadas por la unidad de archivo y para la implementación de nuevas bases de datos es necesario un conocimiento de SQL.

Las bases de datos SQL están soportadas completamente por Android. Para su creación y utilización Android provee de una clase llamada *SQLiteOpenHelper*, que se puede instanciar para crear una base de datos propia. [20]

Una vez creada la base de datos, el funcionamiento es equivalente al funcionamiento normal de las bases de datos de SLQ.

En la aplicación existe una actividad llamada *database*. Su objetivo es mostrar toda la información que ha sido almacenada por la aplicación.

La actividad muestra una lista de todas las entradas de la base de datos de BioAPI junto a sus nombres y su fecha de introducción. Al igual que en la actividad *verify*, se puede acceder a una pantalla individual de cada muestra con la fecha, el nombre y el identificador de cada una. Únicamente se almacenan los BIR procesados, almacenar todas las imágenes no es necesario para el funcionamiento, así que no se puede mostrar la imagen de una huella capturada previamente.

7 Conclusiones y Líneas Futuras

7.1 Conclusiones

El objetivo principal del trabajo se ha cumplido, desarrollando un sistema de identificación biométrica basado en BioAPI utilizando un sensor de huella dactilar que sea capaz de comunicarse de forma inalámbrica, por medio de Bluetooth, con un dispositivo Android.

La realización de esa tarea ha tenido una serie de complicaciones relacionadas con la falta de experiencia y el desconocimiento de ciertas tecnologías. En primer lugar, se trabajaba con un lenguaje de programación y una plataforma en la que no se tenía experiencia, utilizando un estándar (BioAPI) desconocido. Esto ha sido a su vez un estímulo, ya que existía un interés personal en el aprendizaje del funcionamiento de la plataforma Android, pero ha supuesto mucho trabajo. Otro inconveniente ha sido el depender de un dispositivo desconocido, el sensor de huella dactilar, del que se disponía de poca información y soporte. Finalmente estos problemas han sido superados, ganando conocimiento y confianza para proyectos futuros.

El estándar BioAPI es una herramienta muy potente, facilitando el desarrollo y la ampliación de sistemas biométricos, y permitiendo la compatibilidad entre ellos. Esto permite que varios desarrolladores y fabricantes de dispositivos puedan utilizar el mismo sistema de una manera simple. El crecimiento de la tecnología móvil y el renovado interés por las tecnologías de identificación biométrica hacen de BioAPI una ayuda muy útil para desarrolladores que quieran explotar el enorme potencial de las aplicaciones de identificación biométrica móvil.

La utilización de la comunicación inalámbrica extiende las posibilidades de uso de estos sistemas. Aumentan la comodidad y la utilidad, permitiendo que el dispositivo de captura y el dispositivo que procesa y almacena la información estén separados. En la actualidad, con la tendencia a la eliminación de cables en los dispositivos electrónicos, esto supone un gran atractivo.

Como conclusión personal, la elaboración de este trabajo ha supuesto una gran experiencia de aprendizaje. El alumno partía del desconocimiento de las tecnologías a utilizar, el sistema operativo Android, los sistemas biométricos y BioAPI, y no disponía de experiencia en el desarrollo de software complejo de esta magnitud. Gracias a la experiencia adquirida durante este trabajo, se ha ganado confianza y curiosidad por la biometría y el desarrollo de software, ámbitos útiles para el futuro laboral del alumno.

7.2 Líneas futuras

Este trabajo de fin de grado es un ejemplo de sistema biométrico inalámbrico en una plataforma móvil. Es una tecnología en expansión en la actualidad, debido al crecimiento de las tecnologías móviles y los sistemas de seguridad personal. En los últimos años se han desarrollado nuevos dispositivos que utilizan esta tecnología, en particular móviles inteligentes con sensores biométricos integrados, que han ganado popularidad en el uso personal. Esto implica que los sistemas de identificación biométrica son mucho más accesibles y cómodos para el uso personal además del profesional.

El potencial de desarrollo de aplicaciones biométricas móviles es enorme. Puede ser utilizado como método de identificación personal para realizar pagos, confirmar la identidad o desbloquear un dispositivo. No existe necesidad de que el dispositivo que procese la información esté junto al sensor, lo que permite su utilización a distancia, aumentando su seguridad o su utilización por personas discapacitadas con algún tipo de impedimento.

En el ámbito profesional permite nuevos usos y facilidades para los sistemas biométricos. Debido a su pequeño tamaño y su tremenda capacidad de procesado se pueden realizar identificaciones en cualquier lugar de manera fácil y rápida, permitiendo la comunicación con un servidor o una central gracias a la tecnología inalámbrica.

La identificación biométrica en dispositivos móviles es un campo que está creciendo de forma muy destacable en la actualidad, abriendo la puerta a nuevas e intrigantes opciones para esta tecnología.

Bibliografía

[1] CE. "Directiva 2009/136/CE del Parlamento Europeo y del Consejo". http://www.agpd.es/portalwebAGPD/canaldocumentacion/legislacion/union_europea/directivas/common/pdfs/Directiva-136-de-25-noviembre-2009.pdf (5 de Agosto de 2014)

[2] BOE. "Ley orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal". https://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf (5 de Agosto de 2014)

[3] Wikipedia. "Antropometría". http://es.wikipedia.org/wiki/Antropometr%C3%ADa (5 de Agosto de 2014)

[4] Wikipedia. "Biometría". http://es.wikipedia.org/wiki/Biometr%C3%ADa (5 de Agosto de 2014)

[5] SÁNCHEZ REILLO, Raúl. "Identificación Biométrica y su unión con las Tarjetas Inteligentes". http://revistasic.com/revista39/pdf_39/SIC_39_agora.PDF (5 de Agosto de 2014)

[6] LINDOSO MUÑOZ, Almudena. "Contribución al reconocimiento de huellas dactilares mediante técnicas de correlación y arquitecturas hardware para el aumento de prestaciones". Universidad Carlos III. http://e-archivo.uc3m.es/bitstream/handle/10016/5571/Tesis_Almudena_Lindoso_Munoz.pdf?sequence=1 (5 de Agosto de 2014)

[7] Wikipedia. "Huella Dactilar". http://es.wikipedia.org/wiki/Huella_dactilar (5 de Agosto de 2014)

[8] BioAPI Consortium. "History". http://www.bioapi.org/history.asp (5 de Agosto de 2014)

[9] TILTON, C. "BioAPI" http://www.w3.org/2008/08/siv/Slides/Daon/BioAPI-Tilton-2009-full.pdf (5 de Agosto de 2014)

[10] TIOBE. "Index for August 2014 "http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html (5 de Agosto de 2014)

[11] Wikipedia. "Android". http://es.wikipedia.org/wiki/Android (5 de Agosto de 2014)

[12] Pingam. "Bluetooth Fingerprint Reader". http://www.aliexpress.com/item/bluetooth-fingerprint-reader/1425754302.html (5 de Agosto de 2014)

[13] Futronic. "FS28 FIPS201/PIV Bluetooth Fingerprint Scanner". http://www.futronic-tech.com/product_fs28.html (5 de Agosto de 2014)

[14] Smufsbio. "SMUFS-BT RAW Sensor Description and Technical Specification". http://www.smufsbio.com/smufs-bt-raw.aspx (5 de Agosto de 2014)

[15] Toplink Pacific. “Bluetooth Fingerprint Reader General Features, Operations and Characteristics”. http://www.toplinkpac.com/bluefin.html (5 de Agosto de 2014)

[16] Digitalpersona. “TCS1 Large Touch Sensor” http://www.digitalpersona.com/Fingerprint-Biometrics/Embedded-Sensors/TCS1-Large-Touch-Sensor/ (5 de Agosto de 2014)

[17] Toplink Pacific. “BlueFin Sensor Datasheet”. http://www.toplinkpac.com/pdf/BlueFIN%20Brochure.pdf (5 de Agosto de 2014)

[18] National Institute of Standards ans Technology. “NIST Biometric Image Software (NBIS)”. http://www.nist.gov/customcf/get_pdf.cfm?pub_id=51097 (5 de Agosto de 2014)

[19] “Mindtct”. http://ffpis.sourceforge.net/man/mindtct.html (5 de Agosto de 2014)

[20] Android Developers. “SQLiteDatabase”. http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html

Anexo A: Planificación y Presupuesto

A continuación se va a llevar a cabo un desglose de las tareas que se han realizado a lo largo de este trabajo fin de grado, lo que facilitará posteriormente un cálculo aproximado sobre su coste.

A.1 Planificación

Debido a la complejidad de un trabajo de estas características se ha optado por dividirlo en distintas fases, que se van a comentar a continuación:

Fase 1: Documentación inicial

I. Estudio de la plataforma Android y del entorno de desarrollo (20 horas)
II. Preparación de las herramientas de trabajo (4 horas)
III. Búsqueda y realización de tutoriales y aplicaciones sencillas. (30 horas)
IV. Asistencia a charlas y presentaciones sobre Android (16 horas)
V. Asistencia a charlas sobre biometría (5 horas)
VI. Asistencia a charlas sobre BioAPI (5 horas)

Fase 2: Búsqueda y elección del sensor

I. Búsqueda y elección de un sensor de huella dactilar adecuado (20 horas)
II. Estudio del SDK del sensor (15 horas)
III. Pruebas de funcionamiento del sensor (15 horas)

Fase 3: Desarrollo del sistema biométrico

I. Creación de sistema independiente de BioAPI (20 horas)
II. Implementación del SDK del sensor en la unidad de sensor (10 horas)
III. Creación del BSP y sus unidades (30 horas)
IV. Creación de la aplicación (30 horas)

Fase 4: Pruebas

I. Pruebas de campo en un dispositivo Android (10 horas)
II. Corrección y depuración (20 horas)

Fase 5: Elaboración de la memoria

I. Redacción de la memoria (65 horas)
II. Corrección y maquetación (15 horas)

Tabla 1 - Desglose de tareas

FASES	HORAS EMPLEADAS
Documentación inicial	80
Búsqueda y elección del sensor	50
Desarrollo del sistema biométrico	90
Pruebas	30
Elaboración de la memoria	80
TOTAL	**330**

Figura 42: Diagrama de Gantt de la planificación

A.2 Presupuesto del Trabajo Fin de Grado

A.2.1 Costes materiales

Para la realización del trabajo ha sido necesario un ordenador de altas prestaciones, con la capacidad de permitir el correcto funcionamiento del entorno de desarrollo y del dispositivo virtual Android. También se ha necesitado un dispositivo Android con una versión 4.1 del sistema operativo. Considerando un tiempo de amortización de 3 años para ambos dispositivos y teniendo en cuenta la duración del proyecto, los costes quedan expuestos en la Tabla 2. Ha sido necesaria la adquisición de un sensor de huella dactilar Bluetooth compatible con la plataforma Android y la licencia del SDK del sensor, que son mostrados también en la Tabla 2.

Tabla 2 – Costes Materiales

CONCEPTO	PRECIO (€)
Ordenador de altas prestaciones	75,34
Huawei Ascend G510	15,20
Sensor BlueFin y SDK	300,00
TOTAL	**390,54**

A.2.2 Costes de personal

Para la realización de este trabajo, ha sido necesaria la presencia de un jefe de proyecto y un ingeniero.

Tabla 3 – Costes de Personal

OCUPACIÓN	HORAS	PRECIO/HORA	IMPORTE (€)
Jefe de proyecto	25	90	2.250
Ingeniero	305	60	18.300
TOTAL	**330**		**20.550**

A.2.3 Costes totales

Tabla 4 – Costes Totales

CONCEPTO	PRECIO (€)
Costes de materiales	390,54
Costes de personal	20,55
Costes indirectos (20%)	4.188,11
Subtotal	25.128,65
IVA (21%)	5.277,02
TOTAL	**30.405,67**

El coste total del proyecto es de TREINTA MIL CUATROCIENTOS CINCO EUROS CON SESENTA Y SIETE CÉNTIMOS.

Leganés, 15 de septiembre de 2014

El ingeniero